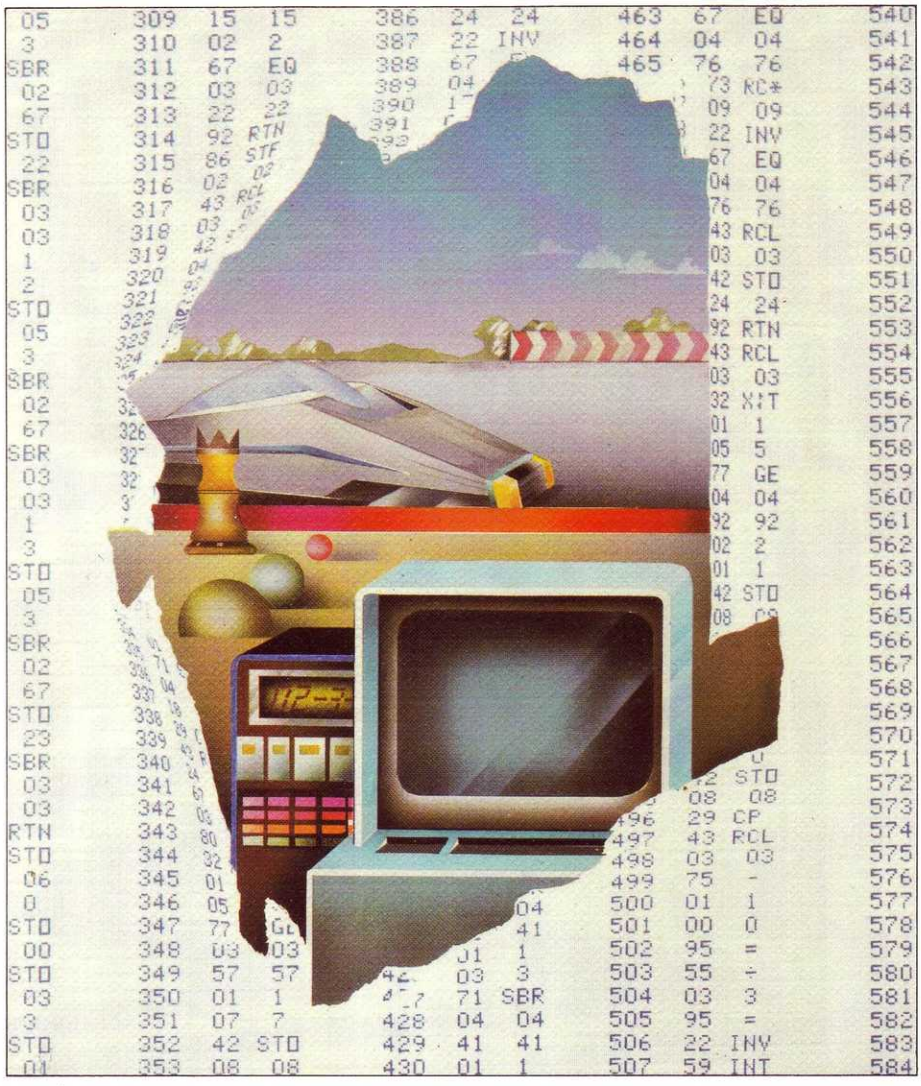
CHIP SPECIAL 2/81

DM 24,--

Eine Sonderpublikation von CHIP. Das Mikrocomputer-Fachmagazin.

Österreich – öS 210.00 Luxemburg – lfr 430.00 Schweiz – sfr 26.00 Niederlande – hfl 30.00



Programme II

für Personal-Computer und Taschenrechner

Programmbeschreibungen, Flußdiagramme und Listings für Spiele, Grafik, Lernen, kaufmännische und naturwissenschaftliche Anwendungen

Über 100 Seiten Programme

CHIP SPECIAL

Die umfangreichen Sonderpublikationen von CHIP

zu den aktuellsten und wichtigsten Themen aus der Welt der Mikrocomputer Technik. Jedes Special für ganze 24 Mark!

Special 1/80:
Mikrocomputer Digest
Das Beste von CHIP über Mikrocomputer-Grundwissen, Programmierung und Programme. Testberichte Personal-Computer und Taschenrechner, Selbstbauanleitungen, Mikrocomputer-Anwendung. Außerdem ein Computerkurs.

Special 4/80:
Mikroprozessoren
Eine komplette Übersicht des Mikroprozessorenangebots. Ausführliche Beschreibung von Daten, Anschlußbelegungen, Innenschaltungen usw. von Speicherbausteinen wie RAMs, ROMs und EPROMs. Dazu die gesamte Systemperipherie.

Special 2/80:
Programme
für Personal-Computer und Taschenrechner, über 100 Seiten Programme mit Beschreibungen, Flußdiagrammen und Listings für Haus, Freizeit und Schule.

Special 1/81:
Hardware-Katalog
für Haus, Freizeit und Beruf. Eine umfassende Übersicht über Einplatinen-Computer, Personal-Computer, Taschenrechner und Drucker der wichtigsten zur Zeit angebotenen Geräte.

CHIP-Specials gibt es
beim Zeitschriftenhandel.
Oder direkt bestellen
beim CHIP-Leserservice

Vogel-Verlag, D-8700 Würzburg 1, Postfach 6740.

CHIP SPECIAL

Eine Sonderpublikation von CHIP. Der Zeitschrift für Mikrocomputertechnik.

Kolumne	5
Was Sie über Programmiersprachen wissen sollten	6

Programme

Programmierhilfen 18

Trace für Superboard/Challenger 1P	18
Cursor für Challenger 1P	20
Repeat-Funktion für CBM	22
Parameter-Übergabe mit dem USR-Befehl für den Exidy-Sorcerer	24
HIR-Register und -Befehle mit dem TI-59/PC 100 C	26
Read Tape and Print	28
Apple II plus Monitorerweiterung	31
Trace für CBM 2001	35
Algebraisches Operationssystem (AOS) für HP-41 C	38
Ausgabe der Symboltabelle für AIM-65/PC 100	41

Spiele 44

Der verflixte Würfel	44
Mühleprogramm für alle Basic-Rechner (32K)	56
Würfelspiel mit dem TI-59/PC 100 C	66
ELIZA II für 6502-Systeme	72

Kaufmännische Anwendungen 78

Prozent-, Zins- und Rentenberechnung für Apple II	78
PC 1211 hilft bei der Nummernsuche	86
Verzinsung mit PC 1211	88
Literaturkartei auf dem CBM 2001	93
Berechnung der Einkommenssteuer für HP 29 C	102
Apple II und Apple II plus als Textsystem	106

Grafik 116

Shapes — Gestalt im Griff	118
Mehr Farbe für den Apple	122
Erweiterter Zeichenvorrat mit dem HP 41 C	123
Dreidimensionale Darstellung auf dem CBM 2001	124
Zeichnen mit CBM	130
Funktionen auf TRS 80	132

Tips und Tricks 134

Kalender für TI-59/PC 100 A	134
CBM 2001 wird vielseitiger	138
Blick in die CPU-Register des CBM 2001	140
CBM-Maschinencode im String gespeichert	141
Hexadezimal-Tastatur mit TI-58	142
Programmverschiebung mit dem TI-59	143
Single Step für Basic-Programme auf dem TRS-80 Level II	149
Basic-Load für CBM	150

Naturwissenschaftliche Anwendungen 152

Taschenrechner-Numerik auf dem TI-59	152
Berechnung von Fourierkoeffizienten für Casio fx-502	156
Basic-Programm zur Berechnung von D/A-Wandler-Fehlern	159
Mengenlehre auf TI-99	162
Wheatstone-Brücke	164
Sternzeit — Sternposition mit dem TI-59	168
TI-59 berechnet Elektronen-konfigurationen der Elemente	173
Mondposition mit TI-58/59	175
Ewiger Kalender auf dem PC 1211	177
Berechnung des Pascalschen Dreiecks auf TI-59	180

Anhang

Buchservice	113
Einkaufsführer	182
Inserentenverzeichnis	186
Impressum	186

Intersoft bietet Software für Z-80/8080 Systeme.

Disketten-Betriebssysteme
CP/M APPLE II (Softcard)
Unter anderem:
Durango F-85
Heath H8 and H89/Zenith Z89
Intel MDS
North Star
OSI C3
TRS-80 Model I, II, III
MP/M for Intel MDS

Festplatten-Anschluß-Module
Unter anderem:
Corvus with Apple II Softcard
Corvus with S100 and TRS-80
Modell II

Systeme Dienstprogramme

Bug and uBug	TRS-80 Model II
Despool	CP/M
Disilog	Customization
Distel	Disk
Edit	Unlock
Edit-80	Word-Master
Filetran	XASM-18
IBM/CPM	XASM-48
Mac	XASM-65
Macro-80	XASM-68
Pasm	XMacro-86
Plink	ZDT
PMate	Z80
Raid	Development
Reclaim	Package
Sid	ZSid

Telekommunikation

BSTAM	BSTMS

Sprachen

ALGOL-60	NEVADA COBOL
APL/V80	PASCAL/M
BASIC-80 (Compiler)	PASCAL/MT
	PASCAL/MT+
BASIC-80 (Interpreter)	PASCAL/Z
	PL/I-80
BDS C Compiler	SMAL/80
CBASIC-2	S-BASIC
CIS COBOL	TINY C
COBOL-80	TINY-C two
FORTRAN-80	W'Smiths C
KBASIC	Compiler
muLISP	XYBASIC
muSIMP	

Sprachen und Anwender-Dienstprogramme

BASIC Utility Disk	MDBS DRS
	MDBS.RTL
Datastar	M/Sort for
Fabs	COBOL-80
Forms 1 for Cis COBOL	PEARL
	QSort
Forms 2 for Cis COBOL	String bit
	String/80
Magsam III	Super-Sort
Magsam IV	Ultrasort II

Textverarbeitung

Magic Wand	Tex
Letterlight	Textwriter III
Microspell	Word-Star
Spellguard	Wordindex

Datenverwaltung

Condor	MDBS.QRS
HDBS	Whatsit?
MDBS	

Anwenderprogramme

CBS	
Selector III-C2	
Selector IV	
Mail-Merge for word star	
Nad	
Postmaster	
Analyst	Statpak
FPL	T/Maker
muMATH	

Datebook
The CP/M Handbook
The C Programming language

Lifelines
Sammelband Lifelines 1–9 (inkl. Febr. 80) ..... 36 DM
Abonnement Lifelines 12 Ausgaben ..... Inland 66 DM
Ausland 80 DM

T/Maker

Dieses leistungsfähige, bildschirm-orientierte Kalkulationsprogramm macht die schwierigsten Kalkulationen zum Kinderspiel. Man definiert die Daten in Zeilen und Spalten, bestimmt die Rechenoperation, setzt die Anfangswerte und der Rest geht von selbst – z.B. wöchentliche Finanzberichte, Vor- und Nachkalkulation von Projekten, Angebotserstellung, usw. Durch Veränderung der Anfangswerte lassen sich beliebige Fälle durchspielen. Die Rechenoperationen umfassen neben den Grundrechenarten Prozent- und Exponentenrechnung, transzendentrale Funktionen, Minima, Maxima und Durchschnittswerte, usw. Dazu ein bildschirm-orientierter, äußerst vielseitiger Editor zum Gestalten von Berichten, Angeboten, usw. Ideal auch für Preislisten, Gebührentabellen, Inventurberichte, Umsatzplanungen.

(erfordert C-BASIC-2) nur 700 DM inkl. Mehrwertsteuer*

C-BASIC-2

Das leistungsfähige kommerzielle BASIC zum günstigen Preis. Festkomma-Variable, Funktionen mit beliebiger Länge und Struktur, vielseitige Befehle zur Datenverwaltung auf Diskette u.v.m.

nur 305 DM inkl. Mehrwertsteuer*

RAID

Werkzeug zur Software-Entwicklung für 8080/8085-Systeme. Enthält Assembler, Disassembler und Debugger. Ermöglicht mehrere Haltepunkte, Anzeigen und Ändern von Registerinhalten, Einzelschrittest, Speicherschutz, Histogramme und Durchsuchen des Speichers.

(erfordert 32K CP/M.) nur 645 DM inkl. Mehrwertsteuer*

* ab Lager München.

Bestellinformation:
Disketten, Bandlaufwerk-Codes, die angegeben werden müssen, wenn Software für die aufgeführten Computer bestellt wird. Alle Software-Produkte haben spezif. Anforderungen an die Hardware oder Software, wie z.B. die CPU, Speichergröße, Betriebssystem oder Prog.-Sprache.
Im Zweifelsfall rufen Sie uns bitte an, um den Format-Code für das jeweilige Gerät abzustimmen.

Altair 8800	B1	Intertec Superbrain DOS 0.1	R7	RAIR Single Density	R9	Shuffleboard	A1
Altos	A1	Intertec Superbrain DOS 0.5	RJ	RAIR Double Density	RE	TRS-80 Model II	A1
Apple CP/M 13 Sector	RG	Intertec Superbrain DOS 3 x	RK	Research Machines 5.25 in	RH	TRS-80 Model III	RU
Apple CP/M 16 Sector	RR	Intertec Superbrain QD	RS	Research Machines 8 in	A1	Vector MZ	Q2
BASF System 7100	RD	Kontron PSI-80	RF	SD Systems 5.25 in	R3	Vector System B	Q2
Cromemco System 3	A1	MITS 3200-3202	B1	SD Systems 8 in	A1	Vista V-80 5.25 in Single Dens	R8
Cromemco Z2D	R6	MSD 5.25 in	RC	Sanyo 7000 5.25 in	RQ	Vista V200 5.25 in Double Dens	P6
Digi-Log Microterm II	RD	Micropolis MOD I	Q1	TEI 5.25 in	R3	Zenith Z89 + Health CP/M	P7
Digital Microsystems	A1	Micropolis MOD II	Q2	TEI 8 in	A1	Zenith A89 + Lifeboat CP/M	P4
Durango F-85	RL	Mostek	A1	TRS-80 Model I STANDARD	R2	Zenith Z89 + Magnolia CP/M	P7
Dynabyte DB8/2	R1	North Star Single Density	P1	TRS-80 Model I + FEC Freedom	RN		
Dynabyte DB8/4	A1	North Star Double Density	P2	TRS-80 Model I + Micromation	A4		
Exidy Sorcerer + Lifeboat CP/M	Q2	North Star Quad Density	P3	TRS-80 Model I + Omikron 5.25 in	RM		
Heath H8 + H17/H27 disk	P4	Ohio Scientific C3	A3	TRS-80 Model I + Omikron 8 in	A1		
Heath H89 + Lifeboat CP/M	P4	Onyx C8001	T2	TRS-80 Model I +			
Heath H89 + Magnolia CP/M	P7	Pertec PCC 2000	A1				
IMS 5000	RA	Quay 500	RO				
IMS 8000	A1	Quay 520	RP				

Bestellkupon:
Senden Sie ☐ gegen Nachnahme ☐ Vorauskasse

____ _____
____ _____
____ _____
____ _____

☐ Senden Sie Ihren Gesamtkatalog
☐ Senden Sie die Kurzbeschreibung für _____
Lieferanschrift: _____

Datum Unterschrift

Einsenden an:
Intersoft GmbH Schloßgartenweg 5, 8045 Ismaning

Lieber Leser,

das Heft, das Sie in Händen halten, hat es mit gleichem Titel schon einmal gegeben. Das war im letzten Jahr. Eine Wiederholung also in der noch jungen und doch bereits so erfolgreichen Reihe der CHIP-SPECIALS. Wiederholung? Die Antwort muß „nein" lauten. Das erste Programm-Heft verkaufte sich wie die berühmten „warmen Brötchen". Indiz dafür, wie sehr die Leser an Programmen interessiert sind. Eigentlich kein Wunder. Sind es doch die Programme, die einen Computer erst das zeigen lassen, was wirklich in ihm steckt. Ohne sie läßt sich das Gerät höchstens als technisches Wunderwerk bestaunen. Oder, in die menschliche Sphäre übertragen, als Wesen, dem der Hauch des Lebens erst eingehaucht werden muß.

Das Interessenspektrum ist bei den Programmen weit gefächert. Sie haben uns die Erfahrungen mit dem Programmservice in der Zeitschrift CHIP und das Leserecho auf das erste SPECIAL „Programme" gelehrt. Wir haben diese Erfahrungen für dieses SPECIAL berücksichtigt.

Das Ergebnis ist ein breitgefächertes Angebot an professionellen Programmen. Manche Zuordnungen zu den einzelnen Rubriken mögen vielleicht auf den ersten Blick etwas willkürlich erscheinen. Warum? Einige Programmbeschreibungen und Flußdiagramme sind so erstellt worden, daß schon fortgeschrittene Anfänger die Programme ohne große Mühen für ihren Personal-Computer oder Taschenrechner umschreiben können. Unter diesem Aspekt ist auch die Zahl der veröffentlichten Programme für die einzelnen Geräte zu sehen. Sie multipliziert sich mit den Möglichkeiten des Umschreibens für andere Geräte.

Viel Spaß wünscht Ihnen
Ihre CHIP-Redaktion

Was Sie über Programmiersprachen wissen sollten

Peter Renner

Wer programmieren will, kommt um eine spezielle Sprache nicht herum. Die Frage ist nur: Welche soll man wählen. Wir haben hier die grundsätzlichen Merkmale der wichtigsten Programmiersprachen zusammengestellt — als Entscheidungshilfe für den Leser.

Betrachtet man die heute an Mikrocomputern gängigen Programmiersprachen, so wird man schnell feststellen, daß deren Wurzeln relativ weit in die Vergangenheit zurückreichen. Es gibt also — zumindest unter den allgemein verfügbaren und gängigen Programmiersprachen — nur relativ wenige, die erst in letzter Zeit realisiert und auch verbreitet wurden.
Klassifiziert man die zur Verfügung stehenden Programmiersprachen weiterhin, so wird man sehr bald feststellen, daß es kaum eine davon gibt, die mehreren Ansprüchen gleichzeitig gerecht wird. Ansprüche in diesem Sinne könnten z.B. sein:

- Einfache Verwendung;
- Möglichkeit der Implementierung komplexer Programmsysteme;
- Einsatz in unterschiedlichen Bereichen.

Aufgrund der unterschiedlichen Anforderungen, die diese speziellen Anwendungstypen an die Programmiersprachen setzen, gibt es auch keine „alleinseligmachende" unter ihnen. Vielmehr wird man immer dazu gezwungen sein, einen gewissen Kompromiß einzugehen, sowohl was die Einfachheit der Programmierung als auch die Kosten des Einsatzes dafür betrifft.

Wir können uns hier nur mit einer kleinen Auswahl allgemein verfügbarer und bekannter Programmiersprachen beschäftigen. Diese Programmiersprachen haben wenigstens eine Gemeinsamkeit: Sie werden an Mikrocomputern entweder bereits eingesetzt, oder man kann das in absehbarer Zeit erwarten. Nicht berücksichtigt sind natürlich Spezialsprachen, wie sie teilweise bei größeren Firmen intern eingesetzt werden, da diese Sprachen nicht frei zur Verfügung stehen (teilweise natürlich auch nicht bekannt werden).

Wie die Programmiersprachen entstanden

Die Historie der Programmiersprachen begann Ende der 50er Jahre mit der Entwicklung der Programmiersprache FORTRAN (**For**mula **Tran**slation). Diese Sprache war nur für mathematisch orientierte Probleme konzipiert worden; wegen dieses eingeschränkten Einsatzbereichs wurde relativ bald darauf die Sprache COBOL (**Co**mmercial **B**usiness **O**riented **L**anguage) entwickelt, deren Anwendungsgebiete vor allem im kaufmännischen Bereich zu sehen sind. Diese beiden Sprachen haben gemein, daß sie relativ ad hoc, den momentanen Bedürfnissen entsprechend, gestaltet wurden. Die Systematik, die späteren Programmiersprachendefinitionen zugrunde liegt, war zum damaligen Zeitpunkt noch nicht bekannt.
Die erste Sprache, die nach solchen systematischen Gesichtspunkten entwikkelt wurde, war ALGOL 60 (**Algo**rithmic **L**anguage). Ihre Sprachelemente wurden zum erstenmal systematisch nach bestimmten Regeln definiert. Ihre Syntax wurde in einer speziellen Form (BNF: Backus-Normal-Form) spezifiziert, und diese exakte Definition erleichtert dem Compilerentwickler die Sprachimplementierung. Ein weiterer Vorteil dieser Sprache gegenüber den vorher genannten ist das sog. Blockkonzept, das es ermöglicht, dynamisch zur Laufzeit speichereffizient zu arbeiten. Ebenso wie bei der Definition der Syntax wurden beim Blockkonzept die inhärenten Probleme systematisch gelöst. Somit stellt diese Sprache den ersten Meilenstein für systematische Entwicklung von Programmiersprachen dar.
Ausgehend von ALGOL 60 wurden die Konzepte der systematischen Definition der Syntax als auch des Blockkonzeptes weiter angewandt. In den 60er Jahren entstanden dann immer mächtigere Sprachen, deren Gipfel Ende der 60er Jahre die beiden Sprachen ALGOL 68 und PL/I (**P**rogramming **L**anguage One) waren. Beide Sprachen haben gemeinsam, daß sie für so gut wie alle Anwendungsfälle geeignet sein sollten und daß die Compilerbauprobleme, die aus der Universalität resultierten, in beiden Fällen nur mangelhaft oder manchmal gar nicht gelöst werden konnten.
Parallelentwicklungen innerhalb der 60er Jahre waren Sprachen, die moderne Konzepte bereits vorwegnahmen, wie z.B. SIMULA 67. Etwa zur gleichen Zeit wurde von McCarthy die Sprache LISP (**L**ist **p**rocessing language) entwickelt, die — wie schon der Name sagt — vor allem zur Manipulation von Listen dient. Trotz dieser Einschränkung hat sich diese Sprache im Einsatz bewährt und wird heute auch an Mikrocomputern verwendet. Parallel zu diesen Entwicklungen innerhalb der 60er Jahre wurden auch andere, etwas spezialisiertere Sprachen entwickelt, wie z.B. BASIC (das wohl als dialogorientiertes FORTRAN anzusehen ist).
Ende der 60er und Anfang der 70er Jahre wurden die Erfahrungen aus den Programmiersprachen SIMULA 67, ALGOL 68 und PL/I systematisch weiterverarbeitet: Es entstand der Wunsch nach möglichst allgemein verwendbaren, jedoch sehr „handlichen" Programmiersprachen. Die erste und mittlerweile am weitesten verbreitete davon ist PASCAL, die Anfang der 70er Jahre entwik-

kelt wurde. Aufgrund der Einfachheit dieser Sprache kann sie nicht den Anspruch vollständiger Universalität wie beispielsweise ALGOL 68 oder PL/I erfüllen; jedoch wurden ihre Konzepte Mitte der 70er Jahre zum Anlaß für Weiterentwicklungen genommen, die mittlerweile immer mehr an Einfluß gewinnen. Ein Beispiel dafür stellt die Sprache ADA dar, deren Entwicklung vom DoD (**D**epartment **o**f **D**efense: Verteidigungsministerium der USA) angestoßen wurde. Sie soll als einheitliche Programmiersprache für verschiedenste Einsatzfälle dienen und wurde 1980 endgültig definiert.
Die Zeit- und Verwandtschaftsbeziehungen einzelner Programmiersprachen — auch wenn sie in der kurzen Aufzählung vorher nicht erwähnt wurden — werden durch Bild 1 dargestellt:

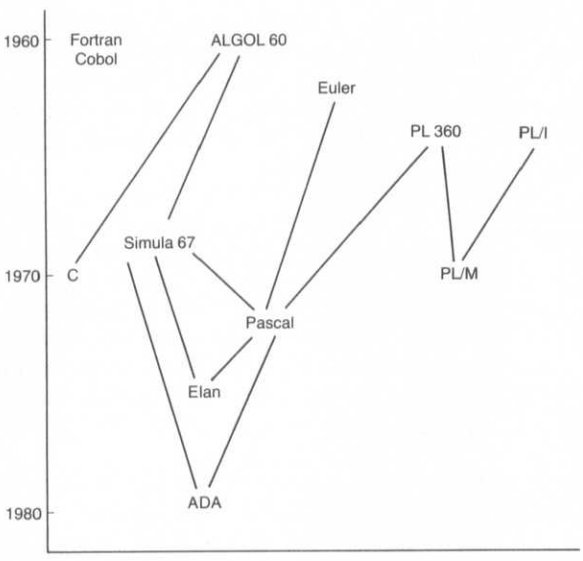


Bild 1: Beziehungen der einzelnen Programmiersprachen

Die speziellen Programmiersprachen

In diesem Kapitel werden drei Programmiersprachen vorgestellt, die zunächst nur für spezialisierte Einsatzfälle entwikkelt wurden. Mittlerweile hat sich jedoch bei diesen Sprachen herausgestellt, daß auch sie für unterschiedliche Anwendungen benutzt werden können. Aus diesem Grund haben sie alle drei bereits eine relativ weite Verbreitung erfahren.

Die Sprache C

Die Sprache C wurde etwa 1970 entwikkelt und gehört zusammen mit BCPL (*B*asic *C*ombined *P*rogramming *L*anguage) und B (Entwickler: Ken Thomson) einer einheitlichen Sprachfamilie an. Ziel war eine Programmiersprache für die PDP 11 unter dem bei Bell Labs verwendeten Betriebssystem UNIX, das zum damaligen Zeitpunkt bereits einen gewissen Standard für benutzerfreundliche Betriebssysteme darstellte. Da es aber in Assembler programmiert war, konnte es keine weitere Verbreitung erfahren.
C wurde in erster Linie als Systemimplementierungssprache realisiert. Mittlerweile hat sich jedoch herausgestellt, daß C auch als Anwendersprache eingesetzt werden kann. Interessant ist die Tatsache, daß bei Bell Labs das dort vorhandene Betriebssystem UNIX später in C umgeschrieben wurde und nur aus diesem Grunde die weite Verbreitung erfahren hat, die es mittlerweile besitzt. Dies läßt den Schluß zu, daß C die Basissprache für portable Systeme darstellen kann.

C war relativ bald auf verschiedenen Mini- und Großcomputern verfügbar: z.B. PDP 11, Honeywell 6000, IBM 370, Interdata 8/32. Ende der 70er Jahre wurde C auch auf verschiedenen Mikrocomputern, u.a. dem Motorola 68 000, implementiert. Außerdem ist die Tatsache interessant, daß erst die Verbindung von UNIX und C zur weiten Verbreitung dieses Pakets beigetragen hat: Im Betriebssystem UNIX ist eine Komponente der C Compiler, der ohne großen Aufwand auf unterschiedlichen Maschinen realisiert werden kann. Nach diesem ersten Schritt kann man das komplette UNIX-Betriebssystem direkt übernehmen und auf die jeweilige Maschine aufsetzen. Dabei ist vor allem die Tatsache interessant, daß für UNIX bereits Hunderte von Anwenderprogrammen geschrieben sind, darunter u.a. verschiedene Editoren, Compiler für andere Programmiersprachen und vieles andere mehr. Durch einen relativ kleinen Schritt — den Kodegenerator für die Sprache C — erhält man dann Zugriff auf sämtliche Komponenten dieses Systems.
Für C existiert im Moment kein international genormter Standard; es ist im Gegenteil sogar an Erweiterungen gedacht, speziell im Hinblick auf den Ausbau des Datentypkonzepts, der bei C im Moment noch zu wünschen übrigläßt. Trotzdem ist der Mangel eines Standards in diesem Falle nicht sehr gravierend, da ja das gesamte UNIX-Paket selbst einen gewissen, in sich geschlossenen Standard darstellt.

Kurzcharakteristik

C besitzt alle üblichen Kontrollkonstrukte, wie z.B.

Fallunterscheidung:
IF — THEN — ELSE
SWITCH (CASE: Mehrfachverzweigung)
Schleifen:
WHILE („pre-check-loop")
DO („post-check-loop")
FOR (Zählschleife)

Diese Kontrollkonstrukte sind bei moderneren Sprachen fast standardmäßig überall enthalten (es gibt natürlich auch einige Ausnahmen, wie die folgenden Kapitel zeigen werden). Es wird bei C jedoch manchmal eine etwas ungewohnte und zum Teil auch schwer lesbare Notation verwendet, die „Umsteigern" das Leben anfangs etwas erschweren kann. Darüber hinaus ist die Vorrangregelung einzelner Operatoren etwas ungewohnt und unterschiedlich zu anderen, gängigen Programmiersprachen.
Als ausgesprochen interessant kann das Pointer-Konzept angesehen werden. Es erlaubt die Adreßmanipulation, ohne Beeinträchtigung der Portabilität. Dies wird dadurch ermöglicht, daß jeder Pointer an einen bestimmten Datentyp gebunden ist. Bei anderen Programmiersprachen, bei denen Pointer nicht an Typen gebunden sind, ist deren Manipulation nur über „normale" arithmetische Operationen oder gar nicht möglich. Werden die arithmetischen Operationen — wie Addition und Subtraktion — verwendet, so wird bei unterschiedlicher Darstellung von bestimmten Datentypen die Portabilität natürlich dadurch beeinträchtigt, daß ja diese Datentypen unterschiedlich auf verschiedenen Maschinen dargestellt werden können, und somit die Addition oder Subtraktion nicht das erwünschte Ergebnis liefern

wird. Bei C mit der strikten Typbindung der Pointer sind Manipulationen durchaus möglich. Dazu folgendes Beispiel:
Wenn ein Pointer P auf das i-te Element eines Arrays mit n Elementen vom Typ x zeigt (Bild 2):

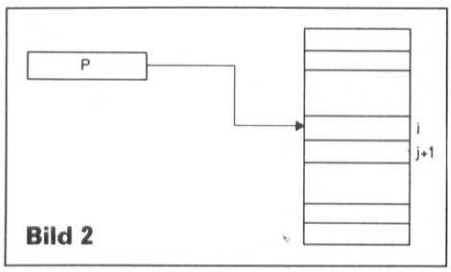


Bild 2

so erhält man das nächste (i+1te) Element durch folgende Operation:

*P+1

Man sieht hier deutlich, daß an der Sprachoberfläche die interne Repräsentation keine Rolle mehr spielt. Aus diesem Grund können auch Programme, die mit Pointers arbeiten, ohne Änderung des Quelltextes auf unterschiedlichen Anlagen ablaufen („oder auf unterschiedliche Anlagen portiert werden").
Das Datentypkonzept von C ist — gegenüber anderen modernen Programmiersprachen — etwas mangelhaft. Dies wirkt sich im wesentlichen dann aus, wenn auf Konsistenz von Typen einzelner Variablen überprüft werden soll. Bei C wird z.B. nicht einmal die Typübereinstimmung zwischen formalen und Aktualparametern bei Prozeduraufrufen überprüft. Dies bedeutet, daß nur die Anzahl der Parameter überprüft wird, nicht jedoch deren Inhalt. So kann beispielsweise eine Prozedur, deren Parameter ganzzahlige Werte — also Integers — sind, mit Characters als Aktualparametern aufgerufen werden. Da natürlich Integer-Operationen auf Characters nicht sinnvoll sind, wird hier sicher ein fehlerhaftes Ergebnis geliefert. Fehler dieser Art können bereits bei der Übersetzung moniert werden, wodurch einige Testläufe vermieden würden. Unter Umständen wird die fehlerhafte Versorgung einer Prozedur mit falschen Daten zur Laufzeit nicht bemerkt, wodurch die Fehlersuche sehr erschwert wird. Zudem wird das interessante Pointerkonzept, das oben erwähnt wurde, dadurch etwas beeinträchtigt, daß es auch erlaubt ist, Integers als Pointers zu verwenden. Dadurch erhält man zwar die Möglichkeit, mit direkten, also absoluten Adressen zu arbeiten, jedoch wird die Arbeit sehr stark dadurch behindert, daß ja hier sowohl ein Portabilitätshindernis eingebaut ist, als auch das Mittel gegeben ist, schwer zu wartende Programme zu erstellen. Betrachtet man dieses Datentypkonzept, so kann man durchaus sagen, daß C für den Anfänger eine ungeeignete Sprache ist. Ein C-Programmierer braucht, bevor er sich damit beschäftigt, relativ viel Erfahrung in der Programmierung.
In neueren Sprachen wird prinzipiell auch die Möglichkeit vorgesehen, Konstante zu benennen. Bei C wird dies durch einen sog. „Pre-Prozessor" realisiert, d.h. einen Vorübersetzer, der nicht in den Compiler eingebaut ist. Er erlaubt es, häufig benötigte Konstanten mit Namen zu benennen, so daß auf diese Weise einfache Änderungen durchgeführt werden können.
Interessant ist die Möglichkeit der Erstellung einer Statistik über Anzahl und Laufzeit einzelner Funktionen. Dies ist speziell für die Optimierung von Programmsystemen von wesentlicher Bedeutung, da bekanntlich auf etwa 10% des Kodes 90% der Laufzeit entfallen, da üblicherweise nur 10% des Programms häufig durchlaufen werden (Problemkern). Es ist normalerweise Aufgabe des Programmierers, genau diese 10% herauszufinden und zeitlich zu optimieren; diese Aufgabe wird durch die Statistik sehr erleichtert.
Ein Mangel der Sprache ist die Ein-/Ausgabe. Die Sprache selbst enthält keine Möglichkeit dazu; sie muß über Prozeduren des Laufzeitsystems realisiert werden. Wegen dieser etwas umständlichen Art führt dies häufig zu unschönen Ergebnissen, da der Typ des ein- oder auszugebenden Datums explizit angegeben werden muß. Änderungen am Programm sind dann natürlich außerordentlich schwierig durchzuführen.

Bewertung

C ist eine sehr interessante Programmiersprache mit weiter Verbreitung. Wegen ihrer Portabilitätskonzepte ist sie vor allem für den Systemprogrammierer geeignet; für den Anwendungsprogrammierer ist sie nur mit Schwierigkeiten verwendbar, da unter anderem die Ein-/Ausgabe schwer zu bedienen ist. Wegen ihrer Einbettung in ein außerordentlich verbreitetes Betriebssystem dürfte C in dieser Umgebung weitere Verbreitung finden. Außerhalb dieser Umgebung ist C so gut wie nicht verbreitet und wird es wohl in absehbarer Zeit auch nicht sein.

Die Sprache FORTH

FORTH wurde von C. H. Moore zwischen 1968 und 1970 entwickelt und ist eigentlich keine Programmiersprache, sondern eher eine Programmiertechnik, die für interaktive Mini- und Mikrocomputerapplikationen angewendet werden kann.
Dies wird am besten dadurch charakterisiert, daß bei FORTH die Systemfunktionen, wie z.B. Programmiersprachenverarbeitung, Betriebssysteme, Anwenderpakete — z.B. zum Editieren oder Debuggen — und die verschiedenen Bibliotheken zusammengefaßt sind. Somit muß der Benutzer auch nur noch mit der Quellsprache arbeiten; es gibt bei FORTH keine Systemaufrufe ähnlich denen anderer Betriebssystemumgebungen.
FORTH ist also ein System, dessen einheitliche, voll integrierte Struktur folgende Komponenten enthält:

- Interaktive, höhere Programmiersprache
- Makroassembler
- Betriebssystem
- Dienstprogramme.

FORTH enthält einige Konzepte, die „normalerweise" nicht an der Programmiersprachenoberfläche zu sehen sind. Dazu gehören z.B.:

- Notierung von Ausdrücken in umgekehrt polnischer Notation
- Stackorganisation.

Diese Konzepte treten bei „normalen" Programmiersprachen nach außen hin nicht in Erscheinung; sie werden üblicherweise durch den Compiler aus dem Quelltext zur effizienten Verarbeitung im Rechner erst erzeugt. Nachteil dieser Stukturen ist es natürlich, daß der Benutzer mit einer etwas unkonventionellen Programmierumgebung arbeiten muß.

Kurzcharakteristik

FORTH besteht aus einer Reihe von modularen, normalerweise unabhängigen Routinen, von denen die meisten in FORTH selbst geschrieben sind. Diese Routinen sind sogenannte „Worte" einer Bibliothek. Diese wird bei der Programmabarbeitung durchsucht, um die „Bedeutung" der einzelnen, eingegebenen Kommandos und Programmanweisungen zu erhalten.
Somit besteht das Programmieren in FORTH nur aus

der Definition neuer Worte

in der oben angeführten Bedeutung. Eine ähnliche Vorgehensweise kann auch bei BASIC angewandt werden. Es gibt aber einen fundamentalen Unterschied in der Abarbeitung von BASIC- und FORTH-Programmen:

- Bei BASIC wird bei der Eingabe das Quellprogramm (meistens) nur minimal verändert, wobei jedem Schlüsselwort nur eine interne Kodierung zugewiesen wird. Also entsteht bei der Programmeingabe ein direktes Abbild des Quellprogramms im Speicher.
- Bei FORTH findet jedoch zusätzlich dazu noch ein weiterer Übersetzungsvorgang statt, bei dem eine Sequenz von Adressen erzeugt wird. Jede dieser Adressen zeigt dann auf eine interne Routine. Diese Adressen können durch einen ausgesprochen schnellen Interpreter abgearbeitet werden. Diese Technik ist nicht FORTH-spezifisch und wird allgemein als „threaded-code" („gefädelter Kode") bezeichnet.

Das Realzeit-Betriebssystem von FORTH ist extrem klein und besteht nur aus einigen hundert Worten. Es kann durch die oben beschriebene Technik des Einfügens neuer oder Weglassens nicht benutzter Worte jeweils den spezifischen Benutzerwünschen angepaßt werden. Hier ist auch der Grund für die Aussage zu sehen, daß FORTH selbst keine Programmiersprache, sondern eher eine Programmiermethodik darstellt, die jedoch durch einige standardmäßig vorgegebene Routinen (Worte) unterstützt wird.

Diesen Vorteil eines enorm kleinen Kerns (minimale Anzahl von Betriebssystem-„Worten"; Interpreter besteht nur aus etwa 500 Kodeinstruktionen) erkauft man sich natürlich mit einigen Nachteilen:

- Bei FORTH existiert kein Anschluß an andere Sprachen. Für Benutzer, die bereits in anderen Programmiersprachen implementiert haben, ist der Übergang zu FORTH demnach nur dann möglich, wenn streng unterschiedliche Anwendungsfälle vorliegen, wobei bereits existierende Teillösungen nicht übernommen werden können.
- Die Kompaktheit führt zu relativ schwer verständlichen Programmen. FORTH-Programme sind im Normalfall allein durch den Quelltext nicht verständlich, da sie nicht den gängigen Gedankenschemata angepaßt sind.

Wie wird nun ein FORTH-Programm abgearbeitet?

Dazu muß man sich veranschaulichen, daß das FORTH-System selbst aus zwei Interpretern besteht, dem sog. Textinterpreter oder äußeren Interpreter und dem inneren Interpreter.

Durch den Textinterpreter wird der eingegebene FORTH-Quelltext in seine interne Form übergeführt, jedes eingegebene Quellwort in der Bibliothek (dem sog. „Dictionary") gesucht und direkt durch den inneren Interpreter ausgeführt.

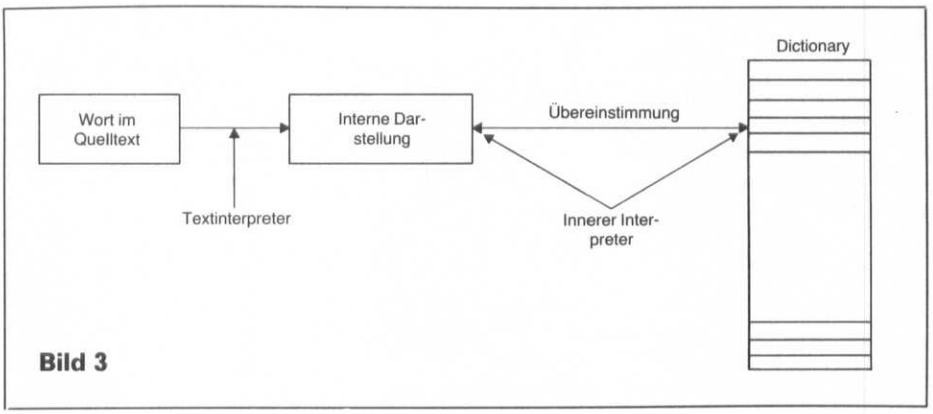


Bild 3

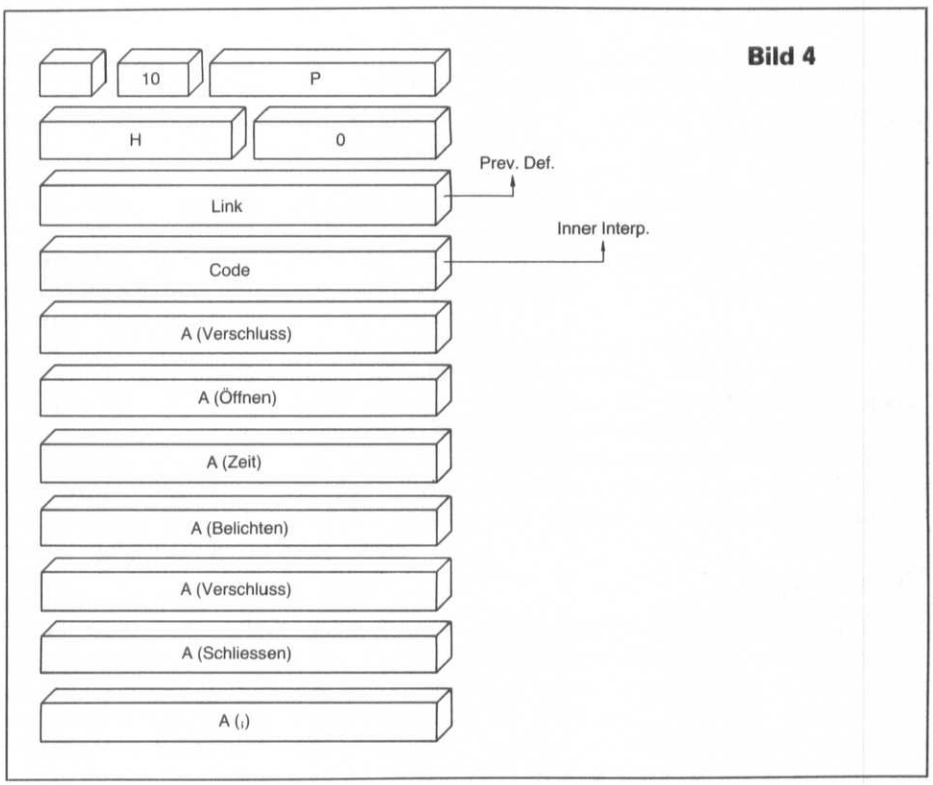


Bild 4

Zur Erläuterung Bild 3.

Der innere Interpreter selbst verarbeitet die obengenannten Worte. Worte entstehen im **Compilemode.** In diesem Modus wird der von außen eingegebene Quelltext nicht direkt ausgeführt, sondern in seine interne Form übergeführt und somit der interne Kode erzeugt, der selbt aus Worten besteht.

Diese nun bereits häufig angeführten Worte sind graphisch in Bild 4 dargestellt.

Der Anfang jedes Wortes enthält dessen Beschreibung, unter anderem auch die Länge. Dies benötigt im Normalfall zwei Speicherzellen. Das nächste Speicherwort besteht aus dem internen Link (Verweis) zum folgenden Programmstatement oder vorhergehenden Wort (verkettete Liste). Das nächste Wort enthält eine Kodeadresse im inneren Interpreter. Im Anschluß daran stehen die einzelnen Adressen der zum Aufbau des aktuellen Wortes benötigten Worte.

Insgesamt wird durch diese Art der Programmierung eine Struktur aufgebaut wie sie in Bild 5 dargestellt ist.

Das dargestellte Beispiel zeigt alle Aktionen, die für eine Fotografie nötig sind:

- Verschluß
- öffnen
- Zeit
- belichten
- Verschluß
- schließen

Jedes dieser Worte kann durch eine Folge von Aktionen, die selbst wieder FORTH-Worte sein können, realisiert werden. Diese Folge realisiert das FORTH-Wort Fotografie.

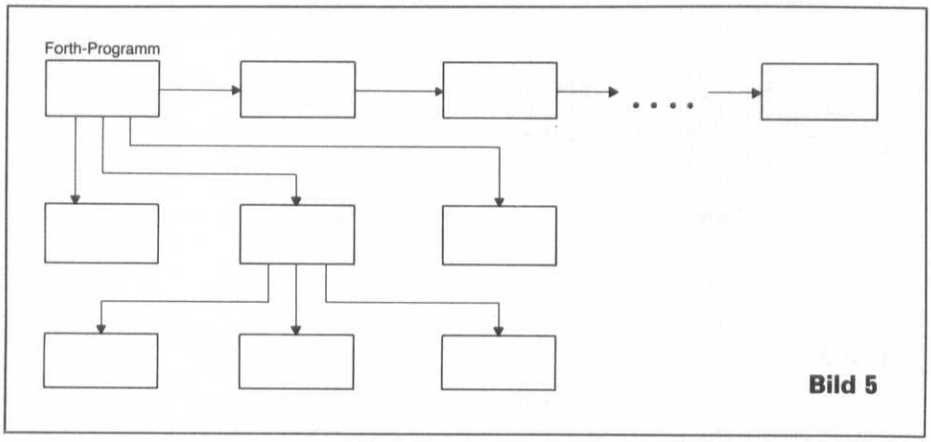


Bild 5

Auf diese Art und Weise wird eine Verkettung der einzelnen FORTH-Worte untereinander erzielt. Dies ist auch gleichzeitig die normale Methode der Vorgehensweise bei der Programmierung: Aus bestehenden Bausteinen werden größere und komplexere dargestellt. Diese Methode wird allgemein die „Bottom-up" Methode der Programmierung genannt.
Bei entsprechender Konzeption kann jedoch auch die heute favorisierte Top-down-Vorgehensweise in FORTH realisiert werden. Dazu verfeinert man die einzelnen FORTH-Elemente immer weiter und erzeugt somit den Strukturbaum von oben nach unten (Top-down).

Bewertung

FORTH ist aufgrund seiner Kompaktheit und leichten Installierbarkeit auf Mikrocomputern relativ weit verbreitet. Ein weiterer wesentlicher Punkt für diese weite Verbreitung dürfte die Tatsache sein, daß jeder Benutzer sich die für ihn spezialisierte Umgebung durch Implementierung neuer FORTH-Statements relativ leicht schaffen kann. So erhält man stets eine benutzerangepaßte Umgebung ohne unbenutzte „Systemleichen" oder Komponenten mit eventuell unpassenden Schnittstellen.
In dieser Erweiterbarkeit liegt jedoch auch eine der großen Schwierigkeiten bei der Anwendung von FORTH: Wegen der Uneinheitlichkeit der zur Verfügung gestellten Bibliotheken müssen bei der Übergabe von FORTH-Programmen jeweils alle benötigten Bibliothekselemente mitgegeben werden. Es genügt nicht — wie bei standardisierten Programmiersprachen — die Übergabe des Quelltextes, da die zugrundeliegenden Funktionen zum Ablauf ebenfalls benötigt werden. Abgesehen von dieser Schwierigkeit gestaltet sich die Übertragung von FORTH-Programmen relativ leicht, da die Schnittstelle zur konkreten Maschine ausgesprochen klein ist.
Insgesamt gesehen ist das Konzept von FORTH so interessant, daß man ihm für längere Zeit ständig wachsende Chancen einräumen kann.

Die Sprache PL/M

Diese dritte unter den etwas spezialisierten Sprachen stellt eine Besonderheit dar: Ihre Spezialisierung betrifft nicht bestimmte Anwendungsgebiete, sondern eine bestimmte Klasse von Mikrocomputern. PL/M ist der Vertreter einer Klasse von Programmiersprachen, die für **einen** bestimmten Mikroprozessor entwickelt wurden (PL/M: Programming Language for Microcomputers). Aufgrund dieser eingeschränkten Zielsetzung ist die vom Namen her implizierte Ähnlichkeit zu PL/I nur recht oberflächlich; die Universalität von PL/I kann natürlich durch eine kleine Sprache nicht erreicht werden, wenngleich die Sprachoberflächen in den identischen Untermengen weitgehend übereinstimmen.
PL/M wurde von der Intel Corporation Mitte der 70er Jahre für den Mikroprozessor 8080 entwickelt. Mittlerweile existiert eine zweite Form — PL/M 86 — für den 16-bit Mikroprozessor 8086.
Grundsätzlich ist zu sagen, daß die Prozessorarchitektur sich voll in der Programmiersprache und deren Operationen widerspiegelt. Aus diesem Grunde ist es in PL/M möglich, fast alle Rechnereigenschaften zu nutzen. Zwischen dem Prozessor und der Programmiersprache gibt es also keine Abstraktion, die durch Datentypen realisiert würde. So ist PL/M ein vollwertiger Ersatz für den Assembler.

Kurzcharakteristik

Wie bereits im vorigen Abschnitt erwähnt, ist PL/M eine höhere Programmiersprache für **einen** Mikroprozessor. Die Sprache gehört somit einer Familie an, die Ende der 60er Jahre durch PL/360 (Entwickler Prof. Wirth; einzusetzen für die IBM 360) initiiert wurde. Heute ist bereits eine Reihe anderer, ähnlich spezialisierter Programmiersprachen in Verwendung. Man denke z.B. an die Zilog-spezifische Programmiersprache PL/Z.
Aufgrund dieser Maschinenorientierung ist es in PL/M möglich, den verwendeten Mikroprozessor 8080 bzw. 8085 direkt durch Programmanweisungen anzusprechen. Es gibt also bestimmte Anweisungen, die es direkt erlauben, spezielle Eigenschaften des Prozessors auszunutzen. Einige dieser maschinenspezifischen Anweisungen werden gar nur auf einen einzigen Maschinenbefehl und nicht auf eine Folge von Maschinenbefehlen abgebildet. Man denke dabei nur an Shift- oder Rotate-Operationen, die ein 8-bit-Datum verschieben bzw. rotieren. Operationen dieser Art sind Maschinenoperationen des 8080 bzw. des 8085.
Trotz dieser Maschinenorientierung ist PL/M eine höhere Programmiersprache; sie erlaubt es, stukturierte Programme zu erstellen, und besitzt auch eine Blockstruktur.
Die zur Realisierung von strukturierten Programmen nötigen Kontrollkonstrukte sind vorhanden:

```
IF, THEN, ELSE
DO WHILE
CASE
...
```

Abgesehen von den zur Stukturierung des Steuerflusses nötigen Kontrollstrukturen erlaubt PL/M auch das Modularisieren von Programmen. Dazu dienen einerseits Funktionen und Prozeduren und andererseits getrennte Übersetzungseinheiten, die später durch einen Binder zusammengefügt werden und erst im Anschluß daran das ablauffähige Programm ergeben.
Als Schnittstelle zwischen den einzelnen Programmoduln dienen dann die Daten. Wie werden diese nun definiert?
Zur Datendefinition dienen die Deklarationen. Jede Deklaration muß mit dem Schlüsselwort „DECLARE" eingeleitet werden. Anschließend daran werden die Namen der Daten definiert und schließlich der Typ. Dieser Typ ist nun nicht zu verwechseln mit einem Datentyp einer anderen, höheren Programmiersprache, wie z.B. bei PASCAL oder ADA. Dieser Typ stellt im wesentlichen nur die Definition dar, wie viele Speicherzellen dieses Datum belegt. Standardmäßig vorgegebene Datentypen sind in PL/M deshalb nur Byte und Word (wobei Word eine Einheit von 2 Byte darstellt).
Durch diese Art der Darstellung ist hier schon die klare Verwandtschaft zur Ma-

schine und nicht zu anderen höheren Programmiersprachen sichtbar.
Es gibt jedoch auch die Möglichkeit, mehrere Daten als Einheit zusammenzufassen. Auch dies ist von anderen Programmiersprachen her bekannt. So ist es beispielsweise möglich, ARRAYs oder auch Strukturen zu definieren (die in anderen Sprachen häufig RECORDs heißen), bei denen die einzelnen Komponenten dieser Struktur durch Namen ausgewählt werden können. Eine Verschachtelung strukturierter Daten ist jedoch nicht möglich.
Die in einem Block definierten Daten sind standardmäßig nur für diesen Block lokal gültig. Da Daten jedoch — wie oben erwähnt — als Schnittstelle zwischen einzelnen Moduln dienen, müssen sie auch in anderen Moduln verwendet werden können, bzw. Daten eines anderen Moduls im momentan gültigen zugreifbar sein. Dazu dienen zwei zusätzliche Attribute, die den Daten bei ihrer Deklaration gegeben werden können: EXTERNAL bzw. PUBLIC. Wird ein Datum als extern definiert, so wird dadurch dem Compiler mitgeteilt, daß dieses Datum erst nach dem Binderlauf adressiert werden kann. Zur Deklaration globaler Daten wird das Attribut PUBLIC vergeben. Das bedeutet, daß andere Moduln, die es verwenden wollen, Zugriff zu seinem Namen und damit auch zu seinem Wert besitzen.
Im Gegensatz zu vielen anderen höheren Programmiersprachen bietet PL/M kein Sprachelement, mit dem Konstante deklariert werden könnten. Dies kann jedoch durch die Verwendung des Wortsymbols „LITERALLY" behoben werden. Durch dieses Wortsymbol wird einem bestimmten Namen ein Text zugewiesen, der vom Compiler bei der Übersetzung substituiert wird. Bei der Anwendung für die Definition von Konstanten könnte dies beispielsweise folgendermaßen aussehen:

```
DECLARE Mehrwertsteuer LITERALLY „13";
```

Im folgenden Programmtext wird vom Compiler jedesmal, wenn der Name Mehrwertsteuer auftritt, dieser Name durch den String „13" ersetzt. Somit gelingt es, Konstante im Programm zu benennen.
Besonders interessant wird diese Anweisung in Verbindung mit der „INCLUDE"-Anweisung. Dadurch können bestimmte Programmteile, die als Datei auf Floppy Disk vorliegen, in das Programm einkopiert werden. Beschäftigen sich nun mehrere Programmierer mit der Realisierung eines Programms, so können allgemein verwendete Daten, Strukturen, Prozeduren oder ähnliches in einer sogenannten INCLUDE-Datei auf Floppy Disk definiert werden. Jeder Programmierer ist dann in der Lage, mittels eines INCLUDE-Statements diese global definierten Daten in sein jeweiliges Programmstück einzukopieren.

So wird die Erstellung von größeren Programmsystemen auch in einer relativ wenig mächtigen Sprache durch sinnvolle Anweisungen unterstützt.

Aus den bisherigen Ausführungen dürfte schon klargeworden sein, daß PL/M weniger eine Anwenderprogrammiersprache als vielmehr eine Systemimplementierungssprache ist, die dem Systemprogrammierer an 8080-, 8085-, 8086- oder 8088-Systemen zur Verfügung steht. Aus diesem Grund ist PL/M auch hauptsächlich an Mikrocomputerentwicklungssystemen wie z.B. Siemens SME oder Intel MDS verbreitet. Die dort existierenden Compiler sind mittlerweile so ausgereift, daß bei den meisten Anwendungen auf Assembler verzichtet werden kann. Beispielsweise sind fast alle Systemprogramme am SME oder MDS selbst in PL/M geschrieben (natürlich auch der PL/M-Compiler).

Bewertung

PL/M war der erste Versuch, theoretische Erkenntnisse des Programmiersprachendesigns auch für Mikrocomputersprachen einzusetzen. Dieser Versuch hat sich in der Praxis sehr bewährt, wie die vielen Anwendungen dieser Sprache zeigen.

Leider existieren in PL/M einige der modernen Spracheigenschaften nicht: Dazu zählt unter anderem die Verwendung von Datentypen. Wegen der Maschinennähe dieser höheren Programmiersprache ist es jedoch möglich, außerordentlich guten Kode (sowohl was die Laufzeit als auch die Größe betrifft) zu produzieren. PL/M stellt eine ausgezeichnete Wahl dar, wenn Produkte an Mikrocomputern entwickelt werden sollen. Dies trifft um so mehr zu, als die Sprache normalerweise in Entwicklungssystemen verwendet wird, an denen außer dem Compiler auch Binder, Lader und Locater vorhanden sind und die entstandenen Programme durch ICE (In-Circuit-Emulator) bzw. ETA (Emulations- und Testadapter) getestet werden können. Bei der Beurteilung dieser Sprache darf man also nicht nur die Sprachoberfläche betrachten, sondern muß deren Einbettung in das komplette Entwicklungssystem berücksichtigen. Einige der bekannten Schwächen werden so für den Anwender weniger gravierend.

Die allgemeinen Programmiersprachen

Dieses Kapitel soll drei weitere Programmiersprachen charakterisieren, die in letzter Zeit vor allem wegen ihrer Modernität im Gespräch sind. Die drei hier angeführten Sprachen — PASCAL, ELAN und ADA — gehören einer einzigen Sprachfamilie an und besitzen so grundsätzlich eine gemeinsame, ihnen allen zugrundeliegende Struktur und Philosophie. Sie sind blockorientiert („ALGOL-artig") und unterstützen sehr stark strukturierte Programmierung.
ELAN und PASCAL sind mittlerweile weit verbreitet; bei ADA kann das für die Zukunft erwartet werden.
Die Zielrichtungen dieser drei Sprachen sind ebenfalls — wie bei den bisher beschriebenen Sprachen — unterschiedlich. Sie wurden zunächst (dies trifft vor allem für ELAN und PASCAL zu) für bestimmte Einsatzfälle konzipiert; es hat sich jedoch herausgestellt, daß alle drei Sprachen aufgrund ihrer Konzeption für unterschiedlichste Einsatzfälle geeignet sind.

PASCAL

PASCAL wurde Anfang der 70er Jahre von Professor Nikolaus Wirth an der Eidgenössischen Technischen Hochschule (ETH) Zürich entwickelt und dort auch zum erstenmal implementiert. Professor Wirth nennt zwei Hauptziele, die er bei der Entwicklung dieser Sprache im Auge hatte:

a) Die Sprache sollte als Lehrsprache für das Programmieren dienen, wobei Programmieren als Disziplin betrachtet wird, die nach systematischen Regeln und Methoden betrieben werden muß.
b) Die Sprache sollte auf konventionellen Rechnern leicht und effizient implementierbar sein.

Diese beiden Ziele wurden bei der Definition von PASCAL nahezu voll erreicht. Es ist nun genau diese Kombination von Eigenschaften, die PASCAL als Meilenstein in der Geschichte der Programmiersprachen erscheinen läßt. PASCAL ist mittlerweile der Ausgangspunkt für eine Menge neuer Programmiersprachen, die meistens viele der Eigenschaften dieser Sprache mitübernehmen konnten.
Wegen der Kompaktheit der Sprache — die eine der Voraussetzungen für die Entwicklung war — kann sie natürlich nicht alle Belange gleichermaßen gut ab-

decken. Sie stellt jedoch wegen ihrer Kompaktheit und der trotzdem sehr hohen Mächtigkeit ein bis dahin nicht erreichtes Optimum an Programmiersprachenmethodik dar.
Im Gegensatz zu den meisten anderen Namen von Programmiersprachen ist PASCAL keine Abkürzung, sondern stellt nur eine Reverenz an den großen Mathematiker Blaise Pascal dar, dessen Vorarbeiten für die Entwicklung heutiger Computer maßgeblich waren.

Kurzcharakteristik

Die beiden bei PASCAL hervorzuhebenden Kernpunkte betreffen:
□ Datentypen und
□ Strukturierte Programmierung.
PASCAL war die erste, allgemein einsetzbare Sprache, die ein völlig in sich konsistentes Datentypkonzept besaß. Was bedeutet nun dies im Klartext?
Unter einem Datentyp versteht man — einfach gesprochen — die Definition möglicher Werte für eine Variable. Diese Definition kann nach unterschiedlichen Kriterien vorgenommen werden. Beispielsweise ist es möglich, für Datentypen, die nur aus wenigen Werten bestehen, diese Werte direkt aufzuzählen. Dabei können die einzelnen Werte durch Namen dargestellt sein:

```
TYPE Farbe = (Rot, Gelb, Grün)
```

Anmerkung: Man beachte, daß die angegebenen Werte **keine** Strings sind, sondern Namen für bestimmte Werte.
Durch diese Anweisung wird ein Datentyp namens Farbe definiert, dessen mögliche Werte nur Rot, Gelb oder Grün sind. Definiert man nun eine Variable dieses Datentyps:

```
VAR Ampel: Farbe
```

so kann dieser Variablen mit dem Namen Ampel nur einer der Werte Rot, Gelb oder Grün zugewiesen werden:

```
AMPEL: = Rot
```

Nach dieser Zuweisung enthält die Variable Ampel den Wert Rot. Zuweisung von Daten anderen Typs an solche Variablen ist nicht zulässig:

```
Ampel: = 1
```

Fehler dieser Art kann schon der Compiler erkennen; sie müssen nicht erst durch Testläufe gefunden werden.
Dieses Konzept der strengen Typbindung von Variablen zieht sich durch die gesamte Sprache.

Eine andere Möglichkeit, solche Aufzählungstypen zu definieren, besteht darin, bestimmte Teilbereiche von bereits vorgegebenen Datentypen auszuwählen. Vorgegebene Bereiche sind unter anderem auch die ganzen Zahlen (INTEGER). Durch die Definition einer unteren und oberen Grenze erhält man dann alle zulässigen Werte eines Datentyps:

```
TYPE Index = 1···10
```

Variable des Datentyps Index können — in diesem Beispiel — nur jeweils einen der Werte 1, 2, ··· 10 annehmen:

```
VAR i : Index
..
i : = 4
```

Zuweisungen wie die folgende

```
i : = 11 oder
i : = 0 oder gar
i : = rot
```

können ebenfalls vom Compiler schon als fehlerhaft erkannt und moniert werden.
Diese Aufzählungstypen stellen zusammen mit den Wahrheitswerten (BOOLEANs, die im Prinzip ebenfalls Aufzählungstypen sind), den Zeichen (CHARACTERs) und den REALs die Basis für alle weiteren Datentypen dar. Versucht man nun komplexere Datentypen aus diesen Basiselementen zusammenzusetzen, so bietet PASCAL verschiedene Strukturierungsmöglichkeiten an. Dazu zählen unter anderem ARRAYs, RECORDs oder SETs.
Unter einem ARRAY versteht man die Ansammlung einer Menge von Daten gleichen Typs, wobei jedes einzelne Element daraus durch einen bestimmten Index ausgewählt werden kann. So besteht das ARRAY aus zwei Datentypen:
□ dem Typ der Elemente und dem
□ Indextyp.
Dieser Indextyp muß selbst wieder ein Aufzählungstyp sein. Für den Typ der einzelnen Elemente gibt es keine Einschränkung; sie können also z.B. Aufzählungstypen oder selbst wieder strukturierte Datentypen sein.
Unter einem RECORD versteht man die Ansammlung verschiedener Elemente, die nicht unbedingt gleichen Typ besitzen müssen. Diese einzelnen Elemente werden durch Namen innerhalb des RECORDs identifiziert. Die Auswahl erfolgt durch Qualifikation: Der Variablenname des RECORDs wird durch den Namen der Komponente „ergänzt". Trennsymbol zwischen den beiden Namen ist ein Punkt. Besteht nun eine Komponente selbst wieder aus RECORDs, so kann diese Methode der Qualifikation fortgesetzt werden.
So bietet PASCAL durch relativ wenige Bausteine die Möglichkeit, ausgesprochen komplexe Datenstrukturen zu definieren und zu verwenden. Beispielsweise sind dadurch Strings (ARRAY OF CHAR) oder andere Strukturen realisierbar.
Unter dem Datentyp SET (Menge) versteht man die ungeordnete Ansammlung von Daten gleichen Typs, wobei die aktuelle Anzahl dieser Elemente nicht bekannt ist. Als Basiselement sind jedoch nur bestimmte Aufzählungstypen zulässig. Für die Verarbeitung von SETs stehen die aus der Mathematik her bekannten Mengenoperationen zur Verfügung.
SETs können die Basis für die elegante Realisierung von Problemen bieten, die sonst nicht trivial zu lösen sind. Betrachtet man die Aufgabe, so lange Zeichen einzulesen, solange das eingelesene Zeichen ein Buchstabe ist, so kann dies in PASCAL durch die Verwendung von SETs relativ einfach realisiert werden: Man definiert sich zunächst einen SET OF CHAR (Menge von Zeichen), die man nur mit Buchstaben vorbesetzt:

```
VAR SetofChar: SET OF CHAR
..
SetofChar := 'A'..'Z'
```

Durch diese Variablendeklaration mit der daran anschließenden Besetzung der Variablen beinhaltet die Menge SetofChar alle Buchstaben. Nun kann die Einleseschleife wie folgt formuliert werden (vorausgesetzt die Variable CH sei vom Typ CHAR):

```
REPEAT
  READ (CH)
UNTIL NOT (CH IN SetofChar)
```

Man sieht, daß hier die Variable CH, deren Wert eingelesen wird, nicht explizit abgeprüft werden muß, ob sie die untere oder obere Grenze der Buchstaben enthält; dies kann ausgesprochen einfach durch die oben angeführte Mengenoperation IN geschehen.
Für die Ein- und Ausgabe stellt PASCAL sequentielle Files zur Verfügung. Diese Dateien werden im Pascalprogramm durch interne Namen angesprochen, die der Benutzer am Programmanfang definiert. Sie besitzen selbst einen bestimmten Elementetyp. Dieser Typ kann einer der vorgenannten — auch komplexen — Datentypen sein. Da die Sprache jedoch keine weiteren Dateikonzepte realisiert (wie z.B. Index-sequentielle Dateien), ist

Jetzt neu!

Intersoft bringt gute Nachrichten für Personal-Computer-Besitzer

Vari-Clean Reinigungskit für Personal-Computer-Lese-/Schreibköpfe und Peripherie

komplett nur 118,–DM inkl. MwSt.

Bestellkupon

Ich/Wir bestellen gegen
○ Vorauskasse ○ Nachnahme

_ Vari-Clean Reinigungskit inkl. 5¼"Reinigungsdiskette 118,– DM ____

_ Vari-Clean Reinigungskit inkl. 8"Reinigungsdiskette 118,– DM ____

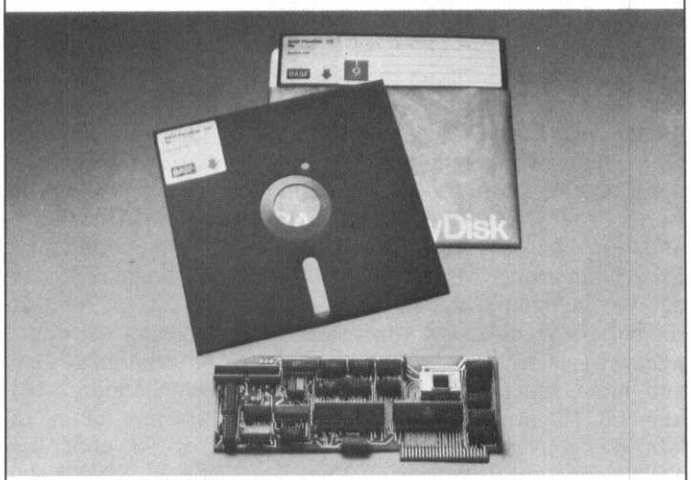
Lieferung/Rechnung an:

Datum Unterschrift

Einsenden an:
Intersoft GmbH, Schloßgartenweg 5
8045 Ismaning

TEACH Systemberatung

Hard- und Software: ALLES aus einer Hand:



Computerhardware:
* Controller f. 8"-Floppylaufwerke f. APPLE II u. ITT2020, 1411 DM. Double Density, Single Density, Double Sided, Single Sided möglich, Dos 3.2 und PASCAL-Kompatibel, bootet von Slot 6, IBM-34 Format (bis zu 6 Laufwerke anschließbar).
* 8"-Floppy-Disk-Laufwerk mit 1 MByte Speicherkapazität in formschönem Gehäuse mit Netzteil, 3783 DM.
* Drucker-Multiplexer-Interface für APPLE II, ITT 2020, mit ihm können mehrere Geräte an 1 Drucker angeschlossen werden, 320 DM.

Wir liefern:
APPLE II, ITT 2020, Pascal-Microengine, DEC, Terminals, Harddisks + Harddiskcontroller, Drucker, Plotter usw.

Computerzubehör:
Terminaltische, Druckertische, Disketten, Druckerpapier, Flachbandstecker und -Kabel, z.B.: Verlängerungskabel für Floppylaufwerke bis zu 3 m, 49 DM.

Computercomponenten:
Floppycontrollersatz WD 1691, WD 2143, FD 1793, 250 DM.
DMA-Controller MC 6844, 99 DM.

TEACH SOFTWARE:
(BASIC-Programme für APPLE II/ITT 2020)
Fakturierung, Lieferscheinschreibung, Auftragswesen, automatisches Bestellwesen, offene Postenliste, Mahnwesen, Kunden-Lieferantenverwaltung, Lagerhaltung (kann mit Finanzbuchhaltung integriert werden.
Finanzbuchhaltung: Komfortable Finanzbuchhaltung mit Bilanzierung. Lohn und Gehalt/Textverarbeitung mit Adreßverwaltung/Angebotswesen mit Textbausteinen, Kundenverwaltung, Artikelverwaltung, Fakturierung, Handwerkerprogramme (schreibt Rechnungen, Angebote mit Stundensätze, Lieferscheine).

Technische Programme:
Schraubenberechnung, Wellenberechnung, Getriebeberechnung, Schulprogramme, Plottprogramme
PASCAL-Programme für APPLE II, ITT 2020, PASCAL-Microengine und allen Computern auf denen das UCSD-PASCAL implementiert ist (z.B.: CPM).
* * Arbeitsvorbereitungsprogramm von FOWIG:
Erstellt die Angebote und deren Kalkulation, erstellt alle Arbeitspapiere wie Arbeitsplan, Lohn, und Materialschein, schreibt die Auftragsbestätigung, überwacht lauf. Aufträge, erst. die Nachkalkulation. * *
* * Architektenausschreibungsprogramm mit Preisspiegel und Massenberechnung, 4900 DM.
* * Dienstleistungsprogramm mit Sammelrechnungen, Tourenplan, Lieferscheine, Kundenverwaltung usw., geeignet für Großküchen, Fensterreinigungen, Gebäudereinigungen usw.
* * Finanzbuchhaltung mit Bilanzabschluß, Summensaldenliste, Kontenverwaltung, Journal, Monatsabschluß usw. * *
* * Formularverarbeitung * *
* * Vertreterprogramm für Handelsvertreter, Versicherungsvertreter, Firmenniederlassungen, verwaltet und überwacht Adressen nach allen Kriterien, schreibt Werbebriefe, Angebote, Rechnungen usw. * *
* * Vertriebsprogramm mit Rückstandslisten, Lagerverwaltung, Kundenverwaltung, Lieferantenverwaltung, Lieferschein- und Rechnungsschreibung usw. * *
* * Textverarbeitungsprogramm mit Adreßverwaltung * *
* * Plottprogramme für Watanabe * *

Angebot des Monats:
APPLE II mit 48 K
2 Floppylaufwerke (5¼")
1 Epsondrucker
1 Textverarbeitungsprogr. m. Adreßverwaltung, **7980 DM** inkl. MwSt.
Bevor Sie Software oder Hardware kaufen, fragen Sie uns. Wir sind nicht billig aber preiswert.

TEACH Hard- und Software Vertrieb GmbH + Coin:
6700 Ludwigshafen-Oggersheim, Alter Frankenthaler Weg,
Telefon (06 21) 67 53 94
7000 Stuttgart-Ditzingen, Siemensstraße 22
7100 Heilbronn, Kernerstraße 62
7191 Schwäbisch Hall, Gelbinger Gasse, Tel. (07 91) 79 04
7500 Karlsruhe, Zähringerstraße 15.

die Ein-/Ausgabe wohl der schwächste Punkt der Sprache. Ohne komplexere Ein- und Ausgabestrukturen ist die Sprache kaum für Aufgaben aus dem kaufmännischen Bereich geeignet.
Der zweite, oben angeführte Kernpunkt der Sprache PASCAL ist die strukturierte Programmierung. Sie zählt unter anderem ebenfalls zur Systematik der Programmierung. Mit PASCAL wurde der Versuch gemacht, den Programmierer nahezu zu einer strukturierten Vorgehensweise zu zwingen. Dies geschieht nun nicht, indem die Sprache einfach kein GOTO (unbedingten Sprung) zuläßt; das wäre schließlich auch nur ein Teilaspekt der strukturierten Programmierung. Vielmehr ist durch die Struktur von PASCAL-Programmen eine Top-down-Vorgehensweise vorgegeben.
Jedes PASCAL-Programm besteht aus zwei Teilen: dem Deklarations- und dem Anweisungsteil. Der Anweisungsteil enthält alle ausführbaren Operationen, unter anderem natürlich auch Aufrufe von Prozeduren oder Funktionen. Im Deklarationsteil werden die benötigten Konstanten, Datentypen und Variablen definiert, aber auch alle Prozeduren und Funktionen, die im Anweisungsteil benötigt werden.
Der Aufbau von Prozeduren oder Funktionen ist analog dem Aufbau von Programmen: Jeder dieser „Blöcke" besteht selbst aus einem Deklarations- und Anweisungsteil, die völlig gleichartig zu den Deklarations- und Anweisungsteilen des Programms aufgebaut sind. Somit kann jeder dieser Blöcke auch selbst wieder Prozeduren und Funktionen enthalten, die lokal zum momentanen Block zu sehen sind. Da der Deklarationsteil vor dem Anweisungsteil notiert wird, enthält er somit auch die Bausteine des Blocks in einer vom Benutzer definierten Reihenfolge.
Durch diese Art der Notation wird der Benutzer gezwungen, sich

- zunächst seine Bausteine zu definieren und dann
- diese Bausteine zu spezifizieren.

Die generelle Vorgehensweise bei der Programmentwicklung ist demnach die Methode der Top-down-Entwicklung; die Methode der Aufschreibung jedoch eine Bottom-up-Methode. Somit wird durch einen relativ einfachen „Trick" eine Synthese dieser beiden Vorgehensweisen hergestellt.

Bewertung

PASCAL zeichnet sich gegenüber vielen anderen Programmiersprachen dadurch aus, daß Unwesentliches aus der Sprache weggelassen wurde, die Sprache also nur aus dem unbedingt notwendigen Kern besteht. Daß dieser Kern trotz seiner geringen Größe sehr mächtig ist, zeigen die vielen Anwendungsfälle: Die Sprache wird allgemein als die heute modernste Mikrocomputer-Programmiersprache angesehen, die bereits im Einsatz ist. Diese Aktualität wäre jedoch ohne die Verwendbarkeit der Sprache in vielen Anwendungsfällen undenkbar gewesen.
PASCAL fehlt zur endgültigen Durchsetzung auf dem Markt leider noch ein international genormter Standard. Dieser Standard wird seit etwa zwei Jahren vorbereitet und dürfte im Laufe des Jahres 1981 endgültig von der ISO (International Standards Organisation) verabschiedet werden.
Wenn man von Schwachpunkten der Sprache PASCAL spricht, so werden häufig die beiden Punkte:

- Ein-/Ausgabe
- Mangelnde Systemprogrammierungseigenschaften

genannt. Diese offensichtlichen Mankos sind der Preis, der für die Einfachheit der Sprache zu bezahlen ist.
PASCAL ist mittlerweile auf allen gängigen Mikrocomputern implementiert: Dazu zählen unter anderen 8080, 8085 und 8086, Z80, 6502, 6800, 68000 usw. Ein Grund für die weite Verbreitung auf Mikrocomputern dürfte in der Tatsache zu suchen sein, daß von der University of California at San Diego von Prof. Ken Bowles das sog. UCSD-PASCAL entwickelt wurde, das nach modernen softwaretechnologischen Gesichtspunkten aufgebaut ist. Dieses PASCAL-System umfaßt nicht nur einen Compiler, sondern beinhaltet vielmehr eine komplette Programmierumgebung, d.h. unter anderem das Betriebssystem, Editor, Compiler, Binder, Lader usw. Da der Großteil dieser Programme selbst in PASCAL geschrieben ist und nur ein geringer Teil — die grundlegenden I/O-Funktionen sowie der Interpreter — im jeweiligen Assembler der Maschine, ist dieses komplette System ohne große Schwierigkeiten auf unterschiedlichen Rechnern installierbar. Abgesehen von dieser einfachen Portierbarkeit demonstriert das System auch den weiten Bereich der Anwendung von PASCAL, wobei gleichzeitig gute Leistungsdaten (wie z.B. Geschwindigkeit, geringer Kodeumfang) erreicht werden können.

ELAN

1974 wurde von Professor C. H. A. Koster an der TU Berlin eine erste, provisorische Beschreibung einer Sprachfamilie mit dem Namen SLAN durchgeführt. Es war das Ziel, eine hierarchische Sprachfamilie zu schaffen, deren einzelne Mitglieder sowohl für den Unterricht als auch für die Systemprogrammierung geeignet sein sollten. Aus dieser Sprachfamilie SLAN wurde die Schul- oder Unterrichtssprache ELAN abgeleitet. Eine erste Implementierung dieser Schulsprache erfolgte im Rahmen von zwei Diplomarbeiten von J. Liedtke und U. Bartling.
ELAN ist also in erster Linie zu Ausbildungszwecken bestimmt. Da die erste Implementierung zu einer Zeit stattfand, als noch an keine Implementierung auf Mikrocomputern zu denken war, fanden diese ersten Implementierungen noch auf Großrechnern wie z.B. Siemens 4004, IBM 370 oder Telefunken TR 440 statt. Später entstand eine Implementierung auf dem ZILOG Z80, die auch die Grundlage für die weiteren Diskussionen wurde: Ähnlich wie beim UCSD-PASCAL-System besteht diese Implementierung nicht nur aus dem Compiler, sondern auch aus einem Betriebssystem. Dieses Betriebssystem mit Namen EUMEL ist außerordentlich benutzerfreundlich und trägt dadurch — in der Kombination mit der Programmiersprache — zur Verbreitung bei. Es ist also auch hier wieder der Trend zu leicht portierbaren, in bestimmte Betriebssysteme eingebetteten Programmiersprachen zu sehen, wie früher schon bei C oder in etwas eingeschränkterem Maße auch bei PASCAL. Für ELAN existiert keine internationale Normung; jedoch wurde ein Sprachvorrat definiert, der als Schulsprache in Deutschland eingesetzt werden soll. Hier ist jedoch noch keine Entscheidung zwischen PASCAL oder ELAN gefallen.

Kurzcharakteristik

Wegen der ähnlichen Zielsetzungen wäre hier ein direkter Vergleich mit PASCAL interessant; dieser Vergleich ist aber im Rahmen dieses Beitrags nicht durchzuführen. Es werden also auch hier wieder nur subjektiv ausgewählte Kriterien untersucht.

Bekanntlicherweise ist ein Prinzip der strukturierten Programmierung auch die Vermeidung von Sprüngen an beliebige Programmstellen. Wegen des Ausbildungscharakters werden bei PASCAL diese Sprünge erschwert; bei ELAN sind sie sogar nicht im Sprachvorrat vorhanden: Es gibt also kein GOTO-Statement.

Das Datentypkonzept von PASCAL wurde bei ELAN erweitert: ELAN besitzt sogenannte „abstrakte Datentypen". Darunter versteht man die Kombination eines — ähnlich wie in PASCAL definier-

ten — Datentyps zusammen mit den auf ihm zulässigen Operationen.

Um diese Kombination zu definieren, bedient man sich eines Sprachelements namens PACKET. In diesen Paketen können dann sowohl die Datentypen als auch Konstanten und die nötigen Operationen definiert werden.

Als Beispiel seien die komplexen Zahlen angegeben. Diese werden üblicherweise durch ein Record (eine Struktur) mit zwei Komponenten, einem Real- und einem Imaginärteil, dargestellt. In der Mathematik sind auf diesen Zahlen Operationen wie Addition oder Subtraktion definiert. Diese Zusammenfassung von Typen, Kostanten und Operatoren hat dann in ELAN folgendes Aussehen:

```
PACKET komplex DEFINES COM-
PLEX, complex null, +;
TYPE COMPLEX = STRUCT
(REAL re, im);
COMPLEX CONST complex null ::
COMPLEX: (0.0, 0.0); (*Konstante*)
COMPLEX OP + (COMPLEX
CONST links, rechts):
COMPLEX:
    (links.re + rechts.re,
    links.im + rechts.im)
END OP +;
...
END PACKET komplex
```

In diesem Programmstück wird auch eine weitere Eigenheit von ELAN deutlich: Namen dürfen auch Leerzeichen enthalten, wie das z.B. bei der Konstanten

```
complex null
```

der Fall ist.
Eine weitere Eigenheit wird ebenfalls symptomatisch deutlich: Es gibt in ELAN absolut keine impliziten Typanpassungen. Dies wird ebenfalls bei der Definition der Konstanten „complex null" deutlich, die ja aus einer Struktur mit zwei REALs besteht. Bei der Definition der Konstanten werden diese beiden Werte jeweils mit 0 vorbesetzt; da jedoch eine REAL 0 als Zahl mit einem Dezimalpunkt zu schreiben ist, muß auch hier bei der Deklaration der String 0.0 stehen und die 0 darf nicht als einzelnes Zeichen stehen. Wäre dies der Fall, so würde das vom Compiler als ganzzahlige 0 erkannt, und eine Zuweisung an einen REAL-Wert wäre nicht legal. Dies bedeutet, daß beispielsweise keine Zuweisung eines ganzzahligen Wertes an eine REAL-Variable durchgeführt werden kann.
Dieses Fehlen von impliziten Typkonversionen erleichtert zwar dem Schüler das Verständnis für die unterschiedlichen Datentypen, die auch unterschiedlich behandelt werden müssen; z.B. wird bei anderen Programmiersprachen die Operation + sowohl für INTEGERs, REALs als auch für Kombinationen davon angewandt, wobei jedoch nicht deutlich wird, daß hier jeweils unterschiedliche interne Operationen ausgeführt werden müssen. Auf der anderen Seite ist es natürlich für den Programmierer — der dieses Prinzip einmal verstanden hat —, es in allen Fällen durchzuhalten.

Eine wesentliche Neuerung in Programmiersprachen wird bei ELAN durch die Unterstützung der Verfeinerung bei der Top-down-Entwicklung geboten.

Bei PASCAL wurde darauf hingewiesen, daß im Anweisungsteil des Programms bei der schrittweisen Verfeinerung Prozedur- und Funktionsaufrufe vorkommen, die im Deklarationsteil definiert wurden. Bei ELAN wird diese schrittweise Verfeinerung noch weiter unterstützt:

Es ist möglich, in einem Block unspezifizierte Aktionsnamen anzugeben, die erst im nachhinein genauer spezifiziert werden müssen. Durch die schrittweise Spezifikation von solchen Elementen entsteht dann der komplette Programmtext, wobei jedoch — und das ist neu — das Programm alle Entwicklungsschritte weiter mit im Quelltext enthält. Im Gegensatz zu PASCAL werden also solche Elemente erst verwendet und dann definiert. Hier ist also auch die Notation eines Programmes Top-down-artig.

Bei PASCAL wurde erwähnt, daß jeder Block, d.h. jede Prozedur, Funktion oder Programm weitere Teilblöcke, also Prozeduren oder Funktionen, enthalten kann, die jeweils lokal zur Umgebung gültig sind. Dies ist bei ELAN nicht möglich: ELAN erlaubt keine geschachtelten Prozeduren. Als Vorteil ist anzusehen, daß die Lebensdauer- und Gültigkeitsbereichsproblematik von Namen auf ein Minimum begrenzt wird. Überlegungen dieser Art belasten den Programmierer nicht.

Es wurde bereits erwähnt, daß ELAN eine strenge Typprüfung durchführt. Damit diese strenge Typprüfung den Programmierer nicht allzusehr belastet, gibt es die Möglichkeit, generische Prozeduren zu definieren. Darunter versteht man die Möglichkeit der Definition von Prozeduren, deren Aktionen auf unterschiedlichen Datentypen ausgeführt werden können.

Schließlich bleibt noch zu bemerken, daß ELAN die Möglichkeit der getrennten Übersetzung von verschiedenen Moduln bietet. Dazu bietet sich das PACKET-Konzept an. Durch die Übersetzung von einzelnen Paketen können ganze Bibliotheken von allgemein verwendbaren Moduln entstehen, die später bei Bedarf in unterschiedliche Programme eingebunden werden können. Beispielsweise kann das Paket „komplex" (wie oben definiert) dann von allen Programmierern benutzt werden, ohne daß es explizit im Programmquelltext des einzelnen Programms miterscheint.

Bewertung

Trotz der eingeschränkten Zielsetzung von ELAN bietet diese Sprache eine Reihe modernster Programmiersprachentechniken. Sie ist somit nicht nur für Programmierschulung, sondern auch für professionellen Einsatz geeignet. Zusammen mit dem unterstützenden Betriebssystem EUMEL wird hier also ein gut verwendbares Paket zur Verfügung gestellt.

Für die Zukunft kann eine weitere Verbreitung dieser Sprache im deutschen Raum erwartet werden. Da jedoch außer der Z80-Implementierung keine weiteren Realisierungen für Mikrocomputer existieren und zudem sowohl der internationale Standard als auch eine internationale Benutzergruppe fehlen, dürfte für absehbare Zeit die Verbreitung nur im deutschsprachigen Raum als Schulsprache gesichert sein.

ADA

ADA unterscheidet sich in vielen Punkten ganz wesentlich von den bisher beschriebenen fünf Programmiersprachen; dazu zählen unter anderem:

- □ Entwicklung
- □ Einsatz
- □ Unterstützung

dieser Programmiersprache. Alle bisher besprochenen Programmiersprachen wurden an einer Stelle entwickelt und dann weiter verbreitet. Bei ADA war die Entwicklung durch ganz andere Parameter gekennzeichnet: Grundvoraussetzung für die Entwicklung war die Tatsache, daß das Department of Defense (DoD: Verteidigungsministerium der USA) nach einer Programmiersprache suchte, mit der „embedded systems" — vielleicht übersetzbar mit „autarke Systeme" — programmierbar sein sollten. Die Entscheidung zur Entwicklung einer neuen Programmiersprache fiel Mitte der 70er Jahre, als festgestellt wurde, daß keine der bisher beim DoD verwendeten

Sprachen allen Ansprüchen gerecht werden konnte.
Daraufhin wurden diese Ansprüche in Form von „requirements" spezifiziert. Diese Spezifikationsphase durchlief dann verschiedene Schritte, wobei die Spezifikation jeweils „härter" wurde. Aus dieser Zeit sind die verschiedenen Spezifikationsschritte bekannt: Strawman, Woodenman, Tinman, Ironman und Steelman. Die letzten Requirements des Steelman wurden zur Basis einer Ausschreibung (competition), an der sich mehrere Firmen beteiligten. Schließlich wurden in einem ersten Schritt vier Sprachen ausgewählt, die weiter untersucht werden sollten. Diese Sprachen wurden zunächst mit Farbennamen bezeichnet: RED, GREEN, YELLOW und BLUE.
Nach der ersten Untersuchung dieser vier Sprachen blieben als endgültige Alternativen noch die Sprachen RED und GREEN im Wettbewerb. Schließlich „überlebte" nur noch der Entwurf von Jean Ichbiah (Firma CII): GREEN.
Interessant ist die Tatsache, daß alle vier in der Endausscheidung berücksichtigten Sprachen auf PASCAL basierten.
In einem letzten Design-Schritt wurde die Sprache GREEN nochmals überarbeitet, einige Funktionen von RED übernommen und schließlich lag Ende 1980 die endgültige Sprachdefinition vor, die mittlerweile bereits den Standard — der international anerkannt werden soll — für diese Sprache darstellt.
Parallel zur Sprachentwicklung wurden auch Anforderungen an die Entwicklungsumgebung von ADA spezifiziert. Auch diese Sprachumgebung wurde durch stetig sich konkretisierende Anforderungen beschrieben: es entstanden Sandman-, Pebbleman- und schließlich 1980 die Stoneman-Requirements für eine Sprachumgebung. Die Stoneman-Requirements beschreiben eine komplette Umgebung — also nicht nur einen Compiler, sondern die Einbettung dieses Sprachsystems in ein Software-Entwicklungssystem. Dazu zählen unter anderem Editoren, Compiler, Binder Lader, Betriebssystemkomponenten usw.
Lange wurde auch über den Namen dieser Programmiersprache diskutiert, schließlich entschied man sich für ADA: Der Name ADA wurde als Reverenz an Lady Ada Augusta Byron, Countess Lovelace, gewählt. Sie war Assistentin von Charles Babbage, der 1822 eine mechanische Rechenmaschine („difference engine") konstruierte.

Kurzcharakteristik

Es ist außerordentlich schwierig, die Programmiersprache ADA kurz zu beschreiben. Allein die formale Definition, die im Dezember 1980 vorgestellt wurde, ist etwa 350 Seiten lang, das Reference Manual mit rd. 250 Seiten etwas kürzer.

Aus der Fülle der in ADA realisierten Spracheigenschaften sollen deshalb nur einige wenige herausgehoben werden.

Einer der wesentlichsten Aspekte bei ADA ist die getrennte Übersetzbarkeit von verschiedenen Moduln. Dies ist bei anderen Programmiersprachen bereits ebenfalls der Fall, so daß aus Bibliotheken bestimmte Bausteine verwendet werden können, die später zu einem Programm zusammengesetzt werden. Es ist jedoch bei keiner anderen Programmiersprache der Fall, daß die Typkonsistenz und die Typüberprüfung auch über getrennte Übersetzungen hinaus in gleichem Maße gewährleistet werden, wie innerhalb einer Übersetzungseinheit. Dies setzt nun sowohl an der Sprachoberfläche als auch beim unterstützenden Sprachsystem eine Reihe von Maßnahmen voraus, die in ADA ebenfalls realisiert werden mußten.

Das augenscheinlichste Merkmal dazu ist die Aufteilung zwischen der Schnittstellendefinition und der Realisierung des einzelnen Moduls.

Die Schnittstellendefinition enthält alle von innen nach außen gehenden Datentypen und deren Verwendung sowie alle Daten, die „exportiert" werden. Diese Information muß in der Modulbibliothek vorhanden sein, damit beim Einbinden eines Moduls die Konsistenz für Schnittstellen gewahrt werden kann. Über die Realisierung der einzelnen Schnittstellen, Funktionen oder Prozeduren wird in der Schnittstellendefinition nichts ausgesagt. Es kann dem Benutzer im Prinzip egal sein, auf welche interne Art und Weise die verwendeten Pakete realisiert werden. Somit ist die Aufspaltung in Schnittstellendefinition und Realisierungsteil eine Grundvoraussetzung für die sichere Verwendung von Bibliotheken.

Es wird häufig darauf hingewiesen, daß es Ziel ist, durch diese einheitliche standardisierte Sprache allgemein verwendbare Bibliotheken von oft gebrauchten Funktionen zu erhalten, die frei zur Verfügung stehen. Dann ist es die wesentliche Aufgabe des Software-Ingenieurs, die für seine Aufgabe benötigten Teile auszuwählen und mit seiner Programmlogik zu verbinden (Bottom-up).

Ein weiteres wesentliches Ziel bei der Realisierung von ADA war es, die altmodische, unsichere und kostenintensive Assemblerprogrammierung durch die Programmierung in ADA voll abzulösen. Bei den meisten Systemimplementierungssprachen kann dies nicht ganz realisiert werden, da mindestens die unterste Schnittstelle zur Maschine üblicherweise in Assembler formuliert werden muß. Bei ADA gibt es dagegen die Möglichkeit, auch maschinenabhängige Programmteile in ADA zu formulieren.

Diese Teile können im Programm explizit ausgewiesen werden und bei einem Übergang auf eine andere Maschine durch andere maschinenspezifische Teile ersetzt werden.

Rein äußerlich unterscheidet sich die Syntax von ADA nicht allzusehr von der von PASCAL (zumindest was die beiden sprachidentischen Teilmengen betrifft). Dadurch wird ein schrittweise erlernbarer Übergang von PASCAL nach ADA gewährleistet.

Bewertung

Eine Bewertung von ADA fällt ebenso wie eine Zusammenstellung ihrer Eigenschaften sehr schwer. Dies hat vor allem den Grund in der Tatsache, daß es im Moment noch keinen kompletten ADA-Compiler gibt, also bisher auch noch keine konkreten Projekte in ADA realisiert wurden. Sicher ist jedoch die Tatsache, daß ADA eine enorm mächtige Sprache ist, die auch entsprechende Unterstützung bei der Programmierung braucht. Sie ist sicher nur eine Programmiersprache für den professionellen Programmierer.

Wegen der enormen Unterstützung durch das DoD scheint ADA jedoch die größten Chancen für die Zukunft zu besitzen. Dies wird allein schon dadurch deutlich, daß überall auf der Welt verschiedene ADA-Compiler in Bearbeitung sind. Offensichtlich scheint es für große Firmen ein „Muß" zu sein, ADA zu unterstützen.

Auf der Mikrocomputerseite ist wieder ein ähnlicher Ansatz wie bei UCSD-PASCAL sichtbar: Es wird ein ADA-Compiler implementiert, der in einen bestimmten Zwischenkode compiliert, der dann von unterschiedlichen Maschinen interpretierbar sein wird. Abgesehen davon wird es für verschiedene 16-bit-Mikrocomputer Kode-Generatoren geben, so daß keine Effizienzhindernisse auftreten werden. Schließlich bleibt noch zu sagen, daß — ähnlich wie bei PASCAL — eine ADA-Microengine entwickelt wurde. Dieser Rechner interpretiert direkt (d.h. hardwaremäßig) den vom Compiler erzeugten Zwischenkode.

Durch den mikroprogrammierten Interpreter sind enorme Laufzeitverbesserungen zu erwarten.

Bei entsprechender Akzeptanz könnte sich ADA also zu **der** Standardsprache für Mikrocomputer entwickeln.

Das Resümee

In den beiden vorangegangenen Kapiteln wurden sechs unterschiedliche Programmiersprachen vorgestellt. Sie unterschieden sich sowohl von der Art ihrer Entwicklung als auch ihres Einsatzes, ihrer Einsatzkosten oder ihrer Konzeption.

Keine dieser Sprachen kann den Anspruch erheben, für alle Zwecke der Programmierung gleich gut geeignet zu sein.

Dabei kommt ADA dem Ideal, zumindest von ihren Möglichkeiten her, am nächsten. Ihre Anwendung wird jedoch dadurch erschwert, daß sie ausgesprochen komplex ist und, abgesehen davon, im Moment auch noch nicht zur Verfügung steht. Von den anderen fünf vorgestellten Sprachen kann keine deren Universalität erreichen. Trotzdem stellen sie einen gewissen Querschnitt durch die Landschaft der Programmiersprachen für Mikrocomputer dar.

Es bleibt dem Anwender überlassen, welche dieser Programmiersprachen er aufgrund seines Wissensstandes und seiner Einsatzfälle verwenden wird. Dabei stehen die meisten dieser Programmiersprachen nicht in direkter Konkurrenz zueinander, sondern ergänzen sich in ihrem Leistungsumfang.

Eines ist jedoch sicher: Mit zunehmender Leistungsfähigkeit der Mikrocomputer werden auch die Ansprüche an die Software steigen. Deren Qualität wird in Zukunft mehr und mehr die Gesamtqualität von Produkten bestimmen. Aus diesem Grund wird auch die Bedeutung der professionell einsetzbaren Sprachen an Mikrocomputern immer mehr zunehmen.

Die Programmierung wird also sicher nicht einfacher werden; und die alten Informatikerwitze werden wohl nie realisiert werden: Dies sind die beiden Sprachen:

BCR: **B**aue **C**ompiler und **r**echne, sowie
SPL: **S**uche **P**roblem und **l**öse.

Literaturverzeichnis

C: The C Programming Language, D. M. Ritchie, S. C. Johnson, M. E. Lesk, B. W. Kernighan, The Bell System Technical Journal, Vol. 57, No. 6, July-August 1978, pp. 1991–2019

FORTH: The FORTH Approach To Operating Systems, E. D. Rather, C. H. Moore, in Ann. Conference of the ACM 1976, Oct. 20, 21, 22; Houston Texas, Conference Proceedings, pp. 233–240

PL/M: PL/M Programmiersprache, System SAB 8080, Siemens AG

Pascal: Pascal User Manual and Report, K. Jensen, N. Wirth, 2nd corrected Reprint of the 2nd Edition 1978, Springer Verlag

ELAN: ELAN Handbuch, R. Hahn, P. Stock, Studientexte Informatik, Akademische Verlagsgesellschaft Wiesbaden, 1979

ADA: Reference Manual for the ADA Programming Language, Proposed Standard Document, United States Department of Defense, July 1980

Commodore

Floppy Disk 8050
1 MB Speicherkapazität

Olympia ES 100, f. CBM
anschl.-fertig 3990,–

Macro-Assembler 10K
in Maschinensprache 698,–

Alle Preise inkl. MWSt.
6 Monate Garantie

Zander
Sachsentor 26
2050 Hamburg 80
(0 40) 721 32 49
Telex 2 173 067

Textverarbeitungs-Progr. für 3032 676,–

NEU!
Textverarbeitungs-Progr. 4 Plus für 8032 998,–

VisiCalc für CBM 3032/8032 598,–

Pascal-Compiler für CBM 298,–

ISAM Maschinenroutinen für Random-Karteien für CBM 3032 u. 8032 337,–

Fakturierung mit integrierter Lagerhaltung Spitzenprogr. individuelle Änderungen mögl. 3500,–

Finanzbuchhaltung für Kleinbetriebe 898,–

Watanabe Plotter f. CBM 26 x 36 cm 3232,–

Interface zum CBM 328,–

Hochauflösende Graphik für CBM 3032 1128,–

Anschluß zum CBM 224,–
Halterung zum Einbau 33,–

Sonderangebot
Memorex-BASF Minidisketten 10 Stck. 89,–

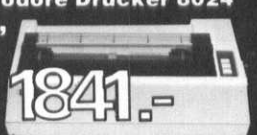
Toolkit im ROM f. 3032/8032 10 zus. Befehle 99,–
Toolkit im ROM f. 8032 27 zus. Befehle 229,–

CBM 4008 1879,–
CBM 4032 2985,–
Sensationspreis

CBM 4016
CBM 3032

Floppy Disk 3040/4040
350 KB.

Commodore Drucker 8024
4498,
4022



CBM 4008 aufgerüstet
auf 16 KB 2210,–
auf 32 KB 2465,–

Preisänderungen vorbehalten.
Leasing möglich.
Wir beliefern auch Händler.

Zander
Sachsentor 26
2050 Hamburg 80
(0 40) 721 32 49
Telex 2 173 067

Qume Sprint 5
Typenraddrucker für Textverarbeitung 7998,–

Centronics 737 Drucker
7 x 8-Matrix, erstklassiges Schriftbild, Einzelblatt u. Endlosformulare 1888,–

Anadex 9500
132-Zeichen-Drucker mit deutschem Zeichensatz 9x9-Matrix. Vor-, Rückwärts-, Parallel- und Serial Interface 200 Zeichen/Sek.

Epson MX 80 FT
80-Zeichen-Drucker
9x9 mit Unterlängen
Deutscher Zeichensatz

Computhink 408 KB für Commodore 2898,–

Umrüstsatz für Umlaute inkl. neues Betriebssystem für CBM 3022 198,–

Umrüstsatz Floppy 3040/4040 Computer 3032/4032 je 229,–

Vieles geht leichter, besser, schneller und effizienter, wenn man sich bestimmter Programmierhilfen bedient. Hier eine Auswahl:
Trace für Superboard/Challenger 1P,
Cursor für Challenger 1P,
Repeat-Funktion für CBM,
Parameter-Übergabe mit dem USR-Befehl für den Exidy-Sorcerer,
HIR-Register und -Befehle für den TI-59/PC 100 C,
Read Tape and Print,
Apple II plus Monitorerweiterung,
Trace für CBM,
Algebraisches Operations-System (AOS) für HP-41 C,
Ausgabe der Symboltabelle für AIM-65/PC 100.

Bernhard Engl

Trace für Superboard/ Challenger 1P

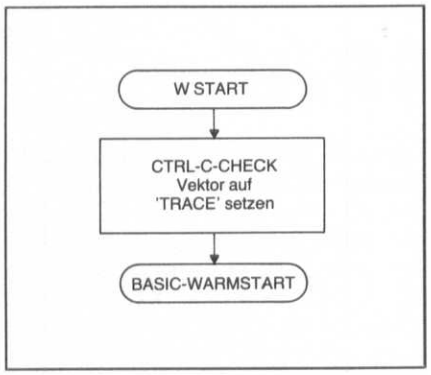
Beschreibung

BASIC-Interpreter für Mikrocomputer waren schon immer mit nur spärlichen Testhilfen ausgestattet. Bei manchen Interpretern, hier dem OSI 8K BASIC, welches als Standardrom in den Personalcomputern Superboard II und Challenger 1P enthalten ist, fehlt sogar eine TRACE-Möglichkeit. Diese einfache Hilfsfunktion ermöglicht es dem Programmierer, den Ablauf seines Programms am Bildschirm zu verfolgen, indem die Nummer der Zeile, die gerade abgearbeitet wird, ausgegeben wird. Fehlt diese Möglichkeit, so wird die Fehlersuche zum Ratespiel, weil man nie genau feststellen kann, was gerade abgearbeitet wird. Um dieses Übel abzuschaffen, muß man versuchen, ein eigenes TRACE-Programm zu schreiben und dieses in den normalen Ablauf des Interpreters einzubauen. Man kann den Teil des Interpreters suchen, der einen Sprung auf eine neue Zeile ausführt und dort einen Uproaufruf einfügen; leider sind hierzu besondere Hardware-Fähigkeiten notwendig; nicht jeder kann ein EPROM brennen und die nötigen Schaltungsänderungen ausführen, um dieses in den Computer einzubauen. Zum Glück gibt

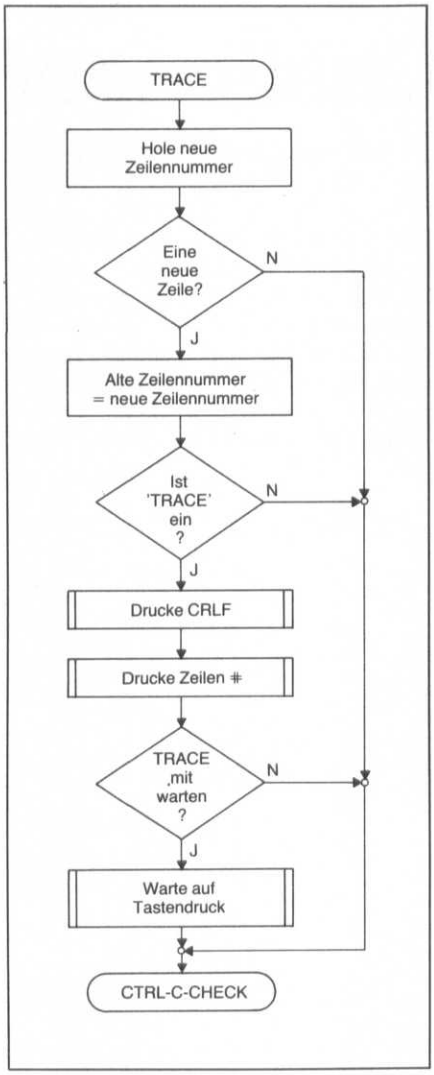
Bild 1:

```
Ø222  A9  2F      WSTART: LDA#  2F     Adresse der
Ø224  A2  Ø2              LDX#  Ø2     TRACE-Routi-
                                       ne . . .
Ø226  8D  1C Ø2           STA   Ø21C   . . . zum CTRL-C-
                                       CHECK-
Ø229  8E  1D Ø2           STX   Ø21D   Vektor.
Ø22C  4C  74 A2           JMP   A274   BASIC-Warmstart.
Ø22F  A5  87      TRACE:  LDA   87     Zeilennummer (L)
Ø231  A6  88              LDX   88     Zeilennummer (H)
Ø233  C5  D8              CMP   D8     Mit der alten Zei-
                                       lennummer ver-
                                       gleichen.
Ø235  DØ  Ø4              BNE   ZNEU   . . . eine neue
                                       Zeile
Ø237  E4  D9              CPX   D9
Ø239  FØ  17              BEQ   CTRLC  . . . die gleiche
                                       Zeile
Ø23B  85  D8      ZNEU:   STA   D8     Alte Zeilennummer
                                       = neue Zeilennum-
                                       mer
Ø23D  86  D9              STX   D9
Ø23F  A5  DA              LDA   DA     Test ob TRACE aus-
                                       geschaltet
Ø241  FØ  ØF              BEQ   CTRLC  ausgeschaltet,
                                       keine Ausgabe
Ø243  48                  PHA
Ø244  2Ø  6C A8           JSR   A86C   Drucke CR/LF
Ø247  2Ø  5A B9           JSR   B95A   Drucke Zeilennum-
                                       mer
Ø24A  68                  PLA
Ø24B  C9  Ø2              CMP#  Ø2     Test ob TRACE mit
                                       Warten
Ø24D  DØ  Ø3              BNE   CTRLC  Trace ohne Warten
Ø24F  2Ø  ED FE           JSR   FEED   Warte auf Tasten-
                                       druck
Ø252  4C  9B FF   CTRLC:  JMP   FF9B   Zum CTRL-C-Test
```

Bild 2:



es eine weitere, einfachere Möglichkeit. Nachdem der Interpreter einen Befehl abgearbeitet hat, ruft er das „CONTROL-C-CHECK"-Upro auf. (Ist CONTROL C gedrückt, so wird der Programmablauf unterbrochen.) Man muß nur den entsprechenden Vektor ändern, so daß nun in die TRACE-Routine gesprungen wird. Hier wird verglichen, ob eine neue Zeile vorliegt. Ist dies der Fall, so wird die Zeilennummer ausgegeben. Dann erfolgt ein Sprung in die CONTROL-C-CHECK-Routine. Die einzige Schwierigkeit bei diesem Verfahren liegt in der Änderung des Vektors. Er wird nämlich nach jedem Warmstart wieder in seinen ursprünglichen Zustand zurückversetzt. Man müßte ihn also jedesmal von BASIC aus mit POKE ändern. Das ist aber nicht möglich, da auch nach einer Immidiate-Mode-Anweisung der Test auf CONTROL C gemacht wird. Dann hat man aber gerade die Hälfte der Adresse geändert, das Resultat ist dann zwar recht interessant, man befindet sich aber wieder in der Startsequenz, von der man wieder nur einen Warm- oder Kaltstart auslösen kann. Hier ist



das Letztere vorzuziehen, wer's genau wissen will, versuche POKE 540,47. Es gibt noch die Möglichkeit, den Vektor im Monitor zu ändern und nachher direkt auf A274 einen Warmstart auszuführen, nur ist dies zeitraubend und unbequem. Besser ist es, den Warmstartvektor auf eine Hilfsroutine zu setzen, die dann ihrerseits den CONTROL-C-CHECK-Vektor ändert. Dann setzt der Monitor den Vektor um, weil aber der Warmstart über den Warmstartvektor läuft und dieser zur Hilfsroutine führt, wird er sofort wieder korrigiert, man hat also nichts mehr zu tun. Das Programm findet man in Bild 1, es besteht aus der obengenannten Hilfsroutine und der eigentlichen TRACE-Routine. Das Flußdiagramm sieht man in Bild 2. Um sich mühevolles Eingeben zu ersparen, wird das BASIC-Hilfsprogramm nach Bild 3 eingegeben und auf Kassette gespeichert. Es wird dann nach jedem Kaltstart geladen und gestartet, dadurch wird das Maschinenspracheprogramm nach Page 2 „gepoket" und der Warmstartvektor geändert. Nachher führt man einen Warmstart aus, weiter ist nichts zu tun. TRACE wird mit POKE 218,1 oder mit POKE 218,2 aktiviert. Im ersten Fall wird nach der Ausgabe der Zeilennummer nicht gewartet. Im zweiten Fall wird gewartet, bis eine Taste gedrückt wird. Mit POKE 218,Ø wird TRACE ausgeschaltet. Dies soll auf alle Fälle vor einem LIST-Befehl getan werden, da auch die LIST-Routine einen CONTROL-C-CHECK durchführt. Ist dann TRACE eingeschaltet, so wird die Ausgabe dermaßen verstümmelt, daß man nichts mehr erkennt. Darüber hinaus entsteht kein Schaden. Das hier verwendete Verfahren hat noch zwei kleine Schönheitsfehler: Nachdem TRACE aktiviert wird, kann die Zeilennummer unbestimmt sein, dann wird eine falsche Zeilennummer ausgegeben. Dies gilt aber nur für die erste, alle folgenden sind fehlerfrei. Der zweite Schönheitsfehler wird durch eine Eigentümlichkeit des BASIC-Interpreters hervorgerufen. Zeilen, welche als einzigen Befehl NEXT, RETURN oder END enthalten, werden nicht ausgegeben, da nach diesen Befehlen kein CONTROL-C-CHEK erfolgt. Diese kleinen Schönheitsfehler mindern aber nicht die Brauchbarkeit des Programms. Der Verfasser verwendet es schon längere Zeit, es hat immer einwandfrei gearbeitet. Die durch die schnellere Fehlersuche gewonnene Zeit läßt sich dann nutzbringender einsetzen, denn gerade Zeit fehlt dem Hobbyisten am meisten.

Bild 3:

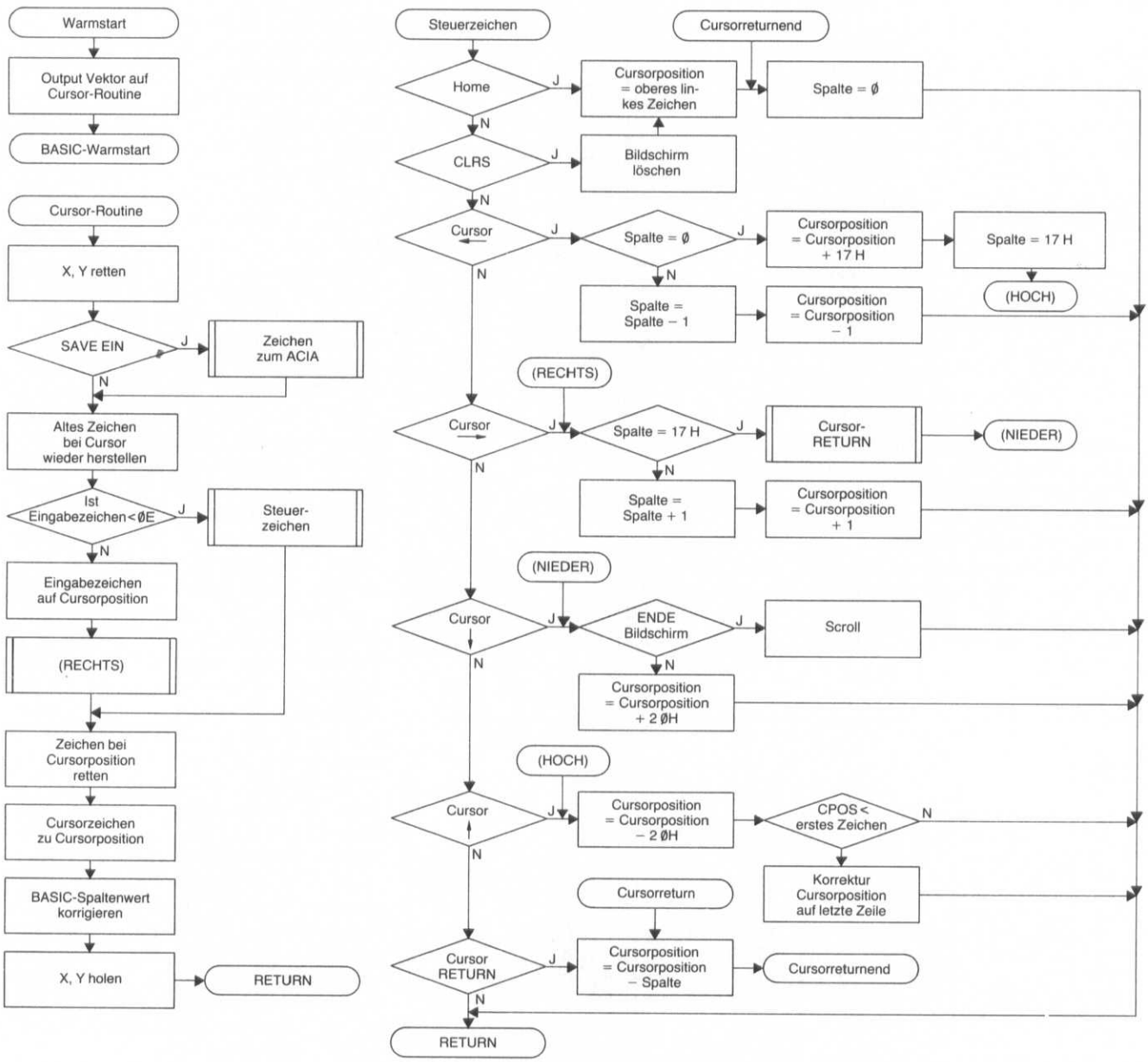
```
 1Ø  REM TRACE
 2Ø  REM WARMSTARTVEKTOR SETZEN UND TRACE AUS
 3Ø  POKE 1,34:POKE 2,2:POKE 218.?
 4Ø  FOR I=546TO 596
 5Ø  READ J:POKE I,J
 6Ø  NEXT :NEW
 7Ø  DATA 169,47,162,2,141,28,2,142,29,2,76,116,162
 8Ø  DATA 165,135,166,136,197,216,2Ø8,4,228,217,24Ø
 9Ø  DATA 23,133,216,134,217,165,218,24Ø,15,72,32
1ØØ  DATA 1Ø8,168,32,9Ø,185,1Ø4,2Ø1,2,2Ø8,3,32,237
11Ø  DATA 254,76,155,255
```

Bernhard Engl

Cursor für Challenger 1P

Beschreibung

Die Computer Challenger 1P und Superboard II verfügen leider über keinen frei beweglichen Cursor. Die einzigen Cursor-Befehle sind Wagenrücklauf und Zeilenvorschub, außerdem steht der Cursor immer auf der letzten Zeile. Der Hersteller hielt einen größeren Komfort nicht für nötig; er schlägt in den mitgelieferten Handbüchern vor, auch komplexere Grafiken mittels „POKE" aufzubauen. Nur ist diese Methode nicht nur langsam, sondern auch nicht gerade sehr freundlich bei der Fehlersuche. Wegen der boshaften Eigenschaft solcher Programme, mittels „POKE" Harakiri zu begehen, ist man gezwungen, nach jeder größeren Änderung eine Bandaufzeichnung zu machen, um die bis dahin geleistete Arbeit nicht zu verlieren. Das kostet dann natürlich viel Zeit. Ferner ist es mit dieser Methode sehr umständlich, Texte zu formatieren oder an irgendeiner Stelle des Bildschirms auszugeben. Daß es fast unmöglich ist, Programme, welche für Computer mit Cursor geschrieben wurden, auf dem Challenger laufen zu lassen, dürfte wohl klar sein. Solch ein Unternehmen läuft dann fast auf ein völliges Neuschreiben hinaus. Um diese Schwierigkeiten aus dem Weg zu räumen und um aus dem Challenger ein



wirklich brauchbares Gerät zu machen, wurde das hier beschriebene Cursor-Programm entwickelt. Es belegt die Adressen DB-E2 in der Zeropage und die Adressen 0222-02F. Da diese Speicherbereiche weder von BASIC noch vom Monitor benutzt werden, geht eigentlich kein Speicherplatz verloren. Das Programm wird nach dem Einschalten des Systems einmalig eingelesen und initialisiert (siehe dazu Tabelle 1). Dann wird ein einmaliger BASIC-Warmstart ausgeführt, und der neue Cursor ist einsatzbereit. Entweder steuert man ihn über die Funktion CHR$(X) oder über einen String, den man vorher mit den Zeichen lädt und

```
0222  A9 2F A2 02 8D 1A 02 8E 1B 02 4C 74 A2 86 E0 84 E1 AC 05 02 F0 04 20 B1
      FC 88 AA A5 DE 91 DB E0 0E B0 06 20 62 02 4C 51 02 8A 91 DB 20 9B 02 B1
      DB 85 DE A9 5F 91 DB A6 E0 A4 DF 84 0E A4 E1 60 A9 02 48 BD 6A 02 48 60
      99 88 99 77 99 99 99 99 C1 9A A9 99 DE 92 A9 20 99 00 D0 99 00 D1 99 00
      D2 99 00 D3 88 D0 F1 A9 A5 85 DB A9 D0 85 DC D0 05 A5 DF 20 CA 02 84 DF
      60 A5 DF C9 17 F0 06 E6 DF A9 01 D0 11 20 93 02 A5 DC C9 D3 D0 06 A5 DB
      C9 5D B0 40 A9 20 18 65 DB 85 DB 90 02 E6 DC 60 A5 DF F0 12 C6 DF A9 01
      85 E2 A5 DB 38 E5 E2 85 DB B0 02 C6 DC 60 A9 17 85 DF 20 B8 02 A9 20 20
      CA 02 A5 DC C9 D0 D0 B0 A5 DB C9 A4 B0 AA A9 D3 85 DC D0 E9 A9 80 8D 00
      02 A9 0A 4C 2D BF
```

dann ausdruckt. Die letzte Methode hat den Vorteil, sehr schnell zu sein. Ein Hinauslaufen des Cursors über die Bildschirmränder ist nicht möglich. Steht er in der obersten Zeile und schickt man ihn nach oben, taucht er auf der untersten Zeile wieder auf. Steht er auf der untersten Zeile und soll nach unten, wird gescrollt. Soll er nach rechts, und er steht am rechten Rand, fährt er an den Anfang der nächsten Zeile, soll er nach links und steht am linken Rand, so fährt er ans Ende der Zeile darüber. Alle möglichen Steuerzeichen sind in Tabelle 2 zusammengefaßt. Die Zeichen von 0 bis 14 (Dezimal), welche keine Steuerzeichen sind, werden leider nicht ausgedruckt. Der begrenzte Speicherplatz machte das unmöglich. Daher wird das Zeichen „UBoot-Heck", welches vom BASIC ausgegeben wird, wenn die Eingabezeile zu lang wird, nicht ausgedruckt. Dies sieht man aber sofort; wenn das Programm stehenbleibt, ist das der Grund und nicht ein Fehler im Cursor-Programm.

Als „Gratis-Zugabe" bekommt man eine Korrektur des Spaltenwertes des Cursors. Der ursprüngliche Cursor tut das nicht, so daß es unmöglich wird, vom BASIC aus per Schleife einen soliden Block auszudrucken. Nach jeweils 72 Zeichen ist für BASIC eine Zeile zu Ende, und es wird CR/LF ausgegeben. Außerdem funktioniert dann die „spaltenmäßige" Ausrichtung ausgedruckter Zahlen nicht richtig. (Mit Komma in der PRINT-Anweisung). Alle diese Fehler werden vom neuen Cursor-Programm automatisch korrigiert.

Wer wissen will, wie der Cursor funktioniert, sei auf das Flußdiagramm (Bild 2) verwiesen. Sollte die Lage des sichtbaren Zeichenfeldes anders sein als bei dem System des Autors, so müssen die entsprechenden Parameter im Programm geändert werden. Ihre Adressen findet man in Tabelle 3. Dabei ist zu beachten, daß man, sobald man die Adresse des linken oberen Zeichens ändert, auch die Adresse des rechten unteren Zeichens in gleicher Weise ändern muß, da andernfalls nicht mehr sichergestellt ist, daß der Cursor richtig funktioniert.

Tabelle 1: Initialisierung des Cursor-Programms. Sie ist von Hand im Monitor vorzunehmen.

Adresse	Inhalt
0001	22
0002	02
00DB	A5
00DC	D0
00DF	00

Tabelle 2: Alle Cursor-Steuerzeichen.

Kode	(HEX)	(DEZ)	Bedeutung
	01	1	Cursor home
	03	3	Bildschirm löschen und Cursor home
	08	8	Cursor nach links
	09	9	Cursor nach rechts
	0A	10	Cursor nach unten (= Zeilenvorschub)
	0C	12	Cursor nach oben
	0D	13	Cursor return (= Wagenrücklauf)

Tabelle 3: Adressen der Parameter

Parameter	Adresse L-Teil	Adresse H-Teil
Adresse des linken oberen Zeichens	028A	028E
Adresse des linken oberen Zeichens −1	02ED	02E7
Adresse des rechten unteren Zeichens +1	02B3	02AD
Anzahl der Zeichen je Zeile	029E, 02D9	—

Robert Loos

Repeat-Funktion für CBM

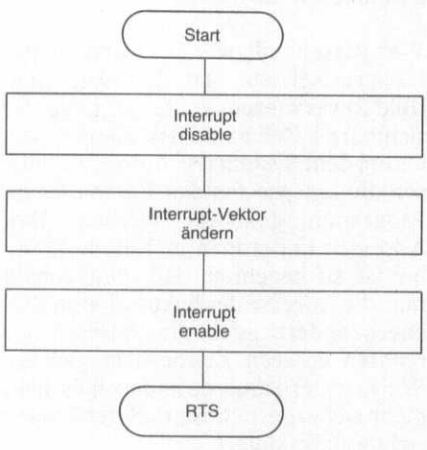
Beschreibung

Dieses Maschinenprogramm bewirkt eine selbständige Wiederholung einer gedrückten Taste nach einstellbarer Zeit und mit einstellbarer Geschwindigkeit. Besonders nützlich wird dies, wenn ein Programm geändert wird und man schnell mit dem Cursor jede beliebige Bildschirmposition erreichen kann, ohne auf den Steuertasten ein mörderisches Stakkato abzuziehen.

Im folgenden möchte ich noch kurz die Wirkungsweise erläutern. Das Programm muß erst eingeschaltet werden. Dies geschieht durch „SYS 826". Die hierbei aufgerufene Routine (Zeilen 3 bis 12) „verbiegt" den Interrupt-Vektor auf die eigentliche Repeat-Routine. Die Befehle SEI am Anfang und CLI am Ende des „Schalters" verhindern einen Interrupt während des Änderns des IRQ-Vektors, was den Rechner zu nicht beeinflußbaren Operationen veranlassen würde.

Die Repeat-Routine selbst fragt zunächst einmal ab, ob und welche Taste im Augenblick gedrückt ist, und vergleicht den erhaltenen Kode mit dem des letzten Interrupts.

Sind diese nicht identisch, wird der neue Kode in 634 abgespeichert, die beiden Zähler für die Anfangsverzögerung (COUNT) und für die Geschwindigkeit (DELAY) zurückgesetzt und die Programmausführung bei der Tastaturroutine (hex E62E) fortgesetzt.



Initialisierung

Sind beide Kodes gleich, d.h. wurde bei den letzten beiden Interrupts die gleiche Taste gedrückt, wird abgefragt, ob der COUNT-Zähler seinen Endwert (hier 60 = 1 s) erreicht hat.

Wenn nicht, wird er um eins erhöht und die Tastaturroutine angesprungen.

Hat er seinen Endwert erreicht, wird die gleiche Prozedur mit dem DELAY-Zähler durchgeführt, der für eine „menschengerechte" Wiederholungsrate sorgt. Hat auch dieser seinen Höchstwert erreicht (hier 5), so wird er zurückgesetzt und in den Tastaturkode Zwischenspeicher (Adresse 151) die Zahl 255 eingeschrieben, was der Tastaturroutine des CBM vorschwindelt, die Taste sei zwischendurch losgelassen worden. Der Effekt ist eine Wiederholung der Taste.

Das Programm wurde auf einem CBM mit neuen ROM geschrieben. Für alte ROM sind folgende Änderungen nötig:

In Zeile 6 muß es heißen STY 21A und in Zeile 8 STY 219 (IRQ-Vektor).

In Zeile 22 LDA 515 sowie in Zeile 45 STX 515 (Tastaturkode-Zwischenspeicher).

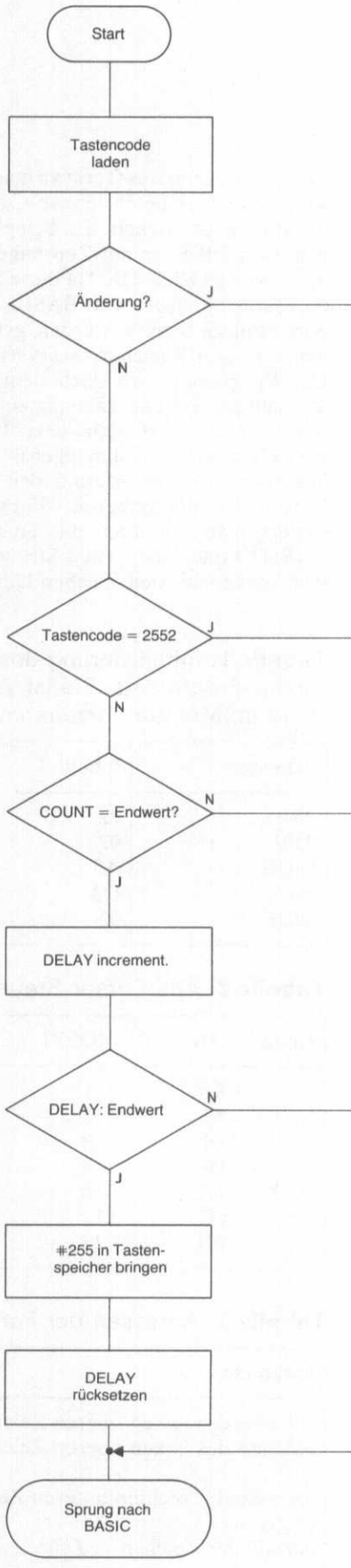
Die Tastaturroutine beginnt bei den alten ROM bei E685, d.h. KEY = E685 in Zeile 20.

Abschließend sei noch gesagt, daß die Repeatfunktion Kassettenoperationen stört. Sie ist vorher durch Drücken von (Shift) + (Run) und anschließendes (Stop) zu neutralisieren.

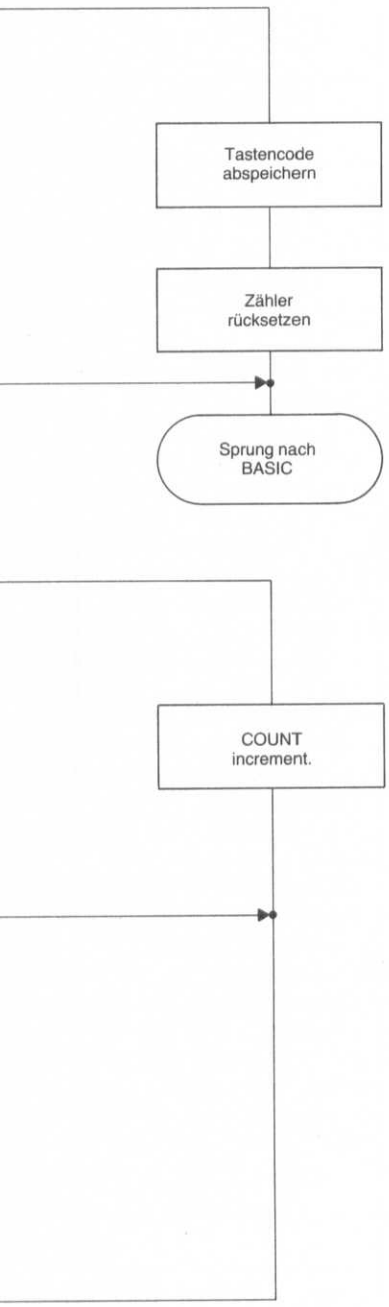
Dadurch wird der alte IRQ-Vektor wieder hergestellt und der Rechner ist wieder im Urzustand. Nach dem Laden kann die Repeatfunktion, sofern durch den Ladevorgang nicht zerstört (von Tape 2 oder direktladendes Maschinenprogramm im 2. Kassettenpuffer), durch SYS 826 wieder aktiviert werden.

Diese Repeatfunktion kann im Gegensatz zu einer bereits anderweitig veröffentlichten auch zusammen mit dem Toolkit verwendet werden.

Für eventuelle Tücken anderer ROM-Versionen sei hiermit jedoch keine Verantwortung übernommen.



Hauptprogramm




```
0382                49 .ENDE

PROGRAMM: REPEAT S

 49 LINES 10 SYMPOLS

033A                 1 *=826.
033A                 2 !
033A  78             3 SEI
033B                 4 !
033B  A0 03          5 LDY #$03
033D  84 91          6 STY $91
033F  A0 45          7 LDY #$45
0341  84 90          8 STY $90
0343                 9 !
0343                10 !
0343  58            11 CLI
0344  60            12 RTS
0345                13 !
0345                14 !
0345                15 *=$0345
027A                16 OLD   =634
027B                17 COUNT =635
027C                18 DELAY =636
027D                19 CODE  =637
E62E                20 KEY   =$E62E
0345                21 !
0345  A5 97         22 LDA 151
0347  CD 7A 02      23 CMP OLD
034A  F0 0E         24 BEQ NULL
034C  8D 7A 02      25 NICHT STA OLD
034F  A9 00         26 LDA #0
0351  8D 7B 02      27 STA COUNT
0354  8D 7C 02      28 STA DELAY
0357  4C 7F 03      29 JMP END
035A                30 !
035A  C9 FF         31 NULL CMP #255
035C  F0 21         32 BEQ END
035E  AE 7B 02      33 LDX COUNT
0361  E0 3C         34 CPX #60
0363  F0 06         35 BEQ REP
0365  EE 7B 02      36 INC COUNT
0368  4C 7F 03      37 JMP END
036B                38 !
036B  EE 7C 02      39 REP INC DELAY
036E  AD 7C 02      40 LDA DELAY
0371  C9 05         41 CMP #5
0373  F0 03         42 BEQ RE
0375  4C 7F 03      43 JMP END
0378  A2 FF         44 RE LDX #255
037A  86 97         45 STX 151
037C  8E 7C 02      46 STX DELAY
037F                47 !
037F  4C 2E E6      48 END JMP KEY
0382                49 .ENDE

OLD   = 027A        COUNT = 027B        DELAY = 027C        CODE  = 027D
KEY   = E62E        NICHT = 034C        NULL  = 035A        REP   = 036B
RE    = 0378        END   = 037F

B*
     PC  IRQ  SR AC XR YR SP
.;  0401 E62E 32 04 5E 00 F8
.
.:  033A 78 A0 03 84 91 A0 45 84
.:  0342 90 58 60 A5 97 CD 7A 02
.:  034A F0 0E 8D 7A 02 A9 00 8D
.:  0352 7B 02 8D 7C 02 4C 7F 03
.:  035A C9 FF F0 21 AE 7B 02 E0
.:  0362 3C F0 06 EE 7B 02 4C 7F
.:  036A 03 EE 7C 02 AD 7C 02 C9
.:  0372 05 F0 03 4C 7F 03 A2 FF
.:  037A 86 97 8E 7C 02 4C 2E E6
.?
.
READY.


 100 REM       ******************
 110 REM       *                *
 120 REM       * REPEAT FUNKTION *
 130 REM       *                *
 140 REM       ******************
 150 DATA 120,160,3,132,145,160,69,132,144,88,96,165,151,205,122,2
 160 DATA 240,14,141,122,2,169,0,141,123,2,141,124,2,76,127,3
 170 DATA 201,255,240,33,174,123,2,224,30,240,6,238,123,2,76,127
 180 DATA 3,238,124,2,173,124,2,201,4,240,3,76,127,3,162,255
 190 DATA 134,151,142,124,2,76,46,230
 200 FOR T=826 TO 897 : READ A : POKE T,A : NEXT : SYS 826
READY.
```

P. Rickmeyer

Parameter-Übergabe mit dem USR-Befehl für den Exidy-Sorcerer

Beschreibung

Oft steht man als Mikrocomputer-Anwender vor dem Problem Daten zwischen Computer und Peripherie bzw. umgekehrt auszutauschen (z.B. AD- und DA-Wandler). Die beiden Programme bieten nun eine Möglichkeit 16 bit + Vorzeichen zwischen BASIC- und Maschinenprogramm und damit Peripherie auszutauschen, ohne umständliche und damit zeitaufwendige Umrechnung in 8-bit-Teilwerte. Und zwar mit dem USR-Befehl.

Die beiden Programme mit den angegebenen ADRn beziehen sich auf den EXIDY SORCERER. Vermutlich arbeiten andere Mikrocomputer den USR-Befehl ähnlich ab. Es muß dann nur die USR-Parameter ADR dem (Z 80) Gerät angepaßt werden.

Mit den angegebenen Programmen kann ein 16-bit-Wert + Vorzeichen ohne Umwandlung in 8-bit-Werte zwischen BASIC- und Maschinen-Programmen ausgetauscht werden.

Üblicherweise wird die USR-Funktion (A = USR (B)) zum Aufrufen von Maschinen-Programmen (MP) verwendet. Den beiden Variablen A, B wird dabei keine Aufmerksamkeit geschenkt. Der USR-Befehl ordnet der Variablen A den Wert des Arguments B zu, d.h., nach dem Rücksprung aus dem MP ist A = B. Diese Eigenschaft läßt sich aber auch sinnvoller ausnutzen.

Floating Point Format Beispiele: (nur 8 bit)

N	Hex	Verschbg.	versch. Hex	ohne „1"	Verschbg. + 80 H.	Fl. Pt.
1	1.	1	.1000	0000	81	81 00
2	10.	2	.1000	0000	82	82 00
3	11.	2	.1100	0100	82	82 40
10	1010.	4	.1010	0010	84	84 20

1. Der USR-Parameter kann während des MPs gelesen werden und z.B. einen DA-Wandler ansteuern oder bei mehreren MP als Sprung-Vektor dienen (JP(HL)).
2. Das MP kann dem USR-Parameter einen anderen Wert zuordnen, der z.B. von einem AD-Wandler kommt.

Grundsätzlich sind diese Funktionen auch mit „POKE" und „PEEK" und vereinbarten Übergabe-ADRn für die 8-bit-Teilwerte möglich. Mit den Unterprogrammen BICON und FLOPT ist diese Parameter-Übergabe aber einfacher. Da ein Wertebereich von 0, ±1, ±2···±65535 zur Verfügung steht, entfällt die Berechnung der 8-bit-Teilwerte und zusätzliche Übergabe-ADRn.

Leider wird der USR-Parameter in einem Floating-Point-Format abgespeichert. Zur Ein- und Ausgabe von der MP-Seite aus muß er also vorher umgewandelt werden. Dies übernehmen die MPs FLOPT und BICON.

FLOPT wandelt eine 16-bit-BIN-Zahl + Vorzeichen in den USR-Parameter. Die inverse Funktion hat BICON. Der USR-Parameter wird in eine 16-bit-BIN-Zahl + Vorzeichen gewandelt.

Das BIN-Komma muß um so viele Stellen nach links geschoben werden, daß die Zahl dann rechts vom Komma steht. Da an dieser Stelle jetzt immer eine „1" steht, wird sie bei positiven Zahlen weggelassen, bei negativen dagegen bleibt sie stehen. Die Verschiebung (N bit) + 80H ergibt den Exponenten. Es steht damit ein Wertebereich von $\pm 2^{-127}/2 \leqq N \leqq \pm 2^{127}$ zur Verfügung.

Die verschobene BIN-Zahl mit der gesetzten oder nicht gesetzten (neg/pos) „1" stellt die Mantisse dar.

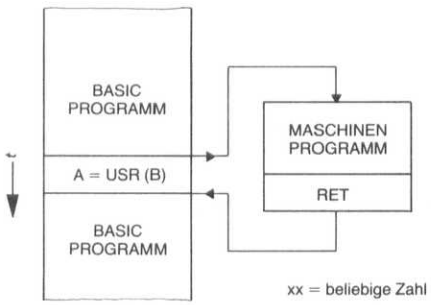
Der USR-Parameter ist im BASIC Control Area 01BF ··· 01C2 abgespeichert. Allerdings im üblichen Z 80-Format. Die höchstwertige Zahl steht in der höchstwertigen ADR.

01BF	01C0	01C1	01C2
	Mantisse		Exponent
z.B. 10	00	00	20	84

≙ 10 D

Von der 3-Byte-Mantisse nutzen die Unterprogramme FLOPT und BICON nur 2 Byte aus.

Zuordnung der Werte während des Programmablaufs.



USR	A	B	USR	A	B
xx	xx	N	xx	xx	N
N	xx	N	N	xx	N
N	xx	N	FLOPT OUT←N	xx	N
			IN←M BICON		
N	N	N	M	M	N
normal USR			USR + Unterprogramme		

```
DU 40 5F

ADDR   0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

0040:  7C 85 28 0C  3E 91 3D 29  30 FC CB 19  CB 1C CB 1D
0050:  22 C0 01 32  C2 01 C9 00  00 00 00 00  00 00 00 00
```

FLOPT-Listing: 16-bit-BIN Zahl + Vorzeichen in Floating Point

```
)DU 60 7F

ADDR   0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

0060:  3A C2 01 6F  67 B7 C8 D6  92 2F 47 0E  80 2A C0 01
0070:  29 CB 11 CB  1C CB 1D A7  10 F9 C9 00  00 00 00 00
```

BICON-Listing: Floating Point Zahl in 16-bit-BIN Zahl + Vorzeichen

```
DU 40 7F

ADDR   0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

0040:  7C 85 28 0C  3E 91 3D 29  30 FC CB 19  CB 1C CB 1D
0050:  22 C0 01 32  C2 01 C9 3A  C2 01 6F 67  B7 C8 D6 92
0060:  2F 47 0E 80  2A C0 01 29  CB 11 CB 1C  CB 1D A7 10
0070:  F9 C9 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

Wenn beide Programme während des USR-Aufrufs benötigt werden, dieses Listing eingeben. Die Programme können beliebig verschoben werden, da nur relative Sprünge vorkommen

```
EXIDY Z-80 ASSEMBLER
   ADDR  OBJECT    ST #

                   0001 ;
                   0002 ;********** BIN / FLOATING POINT WANDLUNG **********
                   0003 ;
                   0004 ;EINE 16BIT BIN-ZAHL + VORZEICHEN WIRD IN EINE
                   0005 ;FLOATING-POINT ZAHL GEWANDELT UND AN DEN USR-
                   0006 ;PARAMETER UEBERGEBEN.
                   0007 ;
                   0008 ;INPUT: 'HL' = 16 BIT BIN-ZAHL
                   0009 ;       'C'  = VORZEICHEN: 0=POS. / 1=NEG.
                   0010 ;
  '0000  7C        0011 FLOPT   LD      A,H
  '0001  85        0012         ADD     A,L              ;ist Zahl = 0 ?
  '0002  280C      0013         JR      Z,NSET-$         ;ja
  '0004  3E91      0014         LD      A,91H            ;Offset-Exponent
  '0006  3D        0015 EXJUS   DEC     A                ;Exponent - 1
  '0007  29        0016         ADD     HL,HL            ;16 Bit links schieben
  '0008  30FC      0017         JR      NC,EXJUS-$
  '000A  CB19      0018         RR      C                ;SIGN setzen
  '000C  CB1C      0019         RR      H                ;Mantisse
  '000E  CB1D      0020         RR      L                ;justieren
  '0010  22C001    0021 NSET    LD      (USR),HL         ;2 Byte Mantisse
  '0013  32C201    0022         LD      (USR+2H),A       ;1 Byte Exponent
  '0016  C9        0023         RET
                   0024 ;
                   0025 ;OUTPUT: 'HL' = 2 BYTE MANTISSE + VORZEICHEN
                   0026 ;        'A'  = 1 BYTE EXPONENT
                   0027 ;         IN USR-PARAMETER 01BFH...01C2H
                   0028 ;
                   0029 ;********** FLOATING-POINT / BIN WANDLUNG **********
                   0030 ;
                   0031 ;EINE 4 BYTE FLOATING POINT ZAHL WIRD IN EINE 16 BIT
                   0032 ;BIN-ZAHL GEWANDELT.
                   0033 ;
                   0034 ;INPUT: USR-PARAMETER 01BFH...01C2H
                   0035 ;
  '0017  3AC201    0036 BICON   LD      A,(USR+2)        ;Exponent
  '001A  6F        0037         LD      L,A
  '001B  67        0038         LD      H,A
  '001C  B7        0039         OR      A                ;  = 0 ?
  '001D  C8        0040         RET     Z                ;ja
                   0041 ;
  '001E  D692      0042         SUB     92H              ;90H - Exp.
  '0020  2F        0043         CPL                      ;  =
  '0021  47        0044         LD      B,A              ;Verschiebung
  '0022  0E80      0045         LD      C,80H
  '0024  2AC001    0046         LD      HL,(USR)         ;Mantisse
  '0027  29        0047         ADD     HL,HL            ;CY = SIGN
  '0028  CB11      0048         RL      C                ;CY=1,'C'=SIGN
  '002A  CB1C      0049 BINJUS  RR      H                ;BIN-Zahl
  '002C  CB1D      0050         RR      L                ;justieren
  '002E  A7        0051         AND     A                ;CY = 0
  '002F  10F9      0052         DJNZ    BINJUS-$
  '0031  C9        0053         RET
                   0054 ;
                   0055 ;OUTPUT: 'HL' = 16 BIT-ZAHL
                   0056 ;        'C'  = VORZEICHEN: 0=POS. / 1=NEG.
                   0057 ;
   )01C0           0058 USR     EQU     01C0H
                   0059 ;
ERRORS=0000
```

Gerd K. Ziran

HIR-Register und -Befehle für den TI-59/PC 100 C

Beschreibung

Als Besitzer eines TI-59/PC 100 C, den ich als mein Hobby betrachte, entwerfe ich nicht nur eigene Arbeits- und Freizeitprogramme, sondern probiere und modifiziere auch fremde Programme, wie sie CHIP Monat für Monat veröffentlicht.

Dabei fiel mir immer wieder ein Befehl auf, den ich in keiner Bedienungsanweisung fand, der HIR-Befehl (Kode 82). Ich benutzte ihn in mehreren fremden Programmen, las auch, daß er auf die sogenannten Hire-Register (von engl.: to hire — mieten) wirken soll, konnte mir aber unter den einzelnen Befehlen nichts vorstellen.

Also machte ich mich mit Hilfe eines Testprogramms an die Entschlüsselung der HIR-Befehle, was mir auch (hoffentlich vollständig) gelang.

Zunächst aber einige Worte über die

HIRE-Register

Der TI-58/59 hat davon acht Stück, die als interne Arbeitsspeicher zum Ablegen von offenen, nicht abgeschlossenen Operationen dienen. Außerdem werden die oberen vier HIRE-Register, HIRE 5 ··· 8, zur Speicherung der Druckkodes verwendet.

Normalerweise sind diese acht HIRE-Register äußerst unzugänglich. Man kann sie zwar indirekt durch Rechen- und Klammeroperationen benutzen oder über Op 1 ··· 4 mit Druckkodes belegen, aber ein direktes und gezieltes Einspeichern und Abrufen von Daten oder gar Speicherarithmetik ist nicht möglich. Die HIRE-Register werden noch nicht einmal von CLR oder CMS gelöscht.

Sie erscheinen also als gegebene, aber nicht direkt nutzbare, interne Arbeitsspeicher.

Das bleiben sie auch, solange über die Tastatur gearbeitet wird. Anders sieht es in Programmen aus, denn hier werden die HIRE-Register zu praktisch verwendbaren Datenspeichern, eben mit Hilfe der HIR-Befehle.

HIR-Befehle

Wie ein HIR-Befehl, etwa HIR 15

Druckoperationen Op Ø1	—	HIRE 5
Op Ø2	—	HIRE 6
Op Ø3	—	HIRE 7
Op Ø4	—	HIRE 8
Op ØØ	—	HIRE 5 ··· 8
Statistische Funktionen $\Sigma +$	—	Werteeingabe; HIRE 7 und 8; HIRE 7: $X_i + 1$; HIRE 8: $X_i * Y_i$
INV $\Sigma +$	—	Eingabekorrektur; HIRE 7 und 8; HIRE 7: $X_{i-1} + 1$; HIRE 8: $-X_i * Y_i$
$\bar{X}$	—	Mittelwert; HIRE 1 HIRE 1: ΣY
INV $\bar{X}$	—	Standardabweichung; HIRE 1, 2 und 8 HIRE 1: $\frac{\Sigma(Y_i - Y_{i-1})^2 + (\Sigma(Y_i - Y_{i-1}))^2}{N}$ HIRE 2: $(\Sigma Y)^2$ HIRE 8: N-1
Op 11	—	Varianz; HIRE 1 und 2 HIRE 1: $-\bar{X}^2$; HIRE 2: ΣX^2
Op 12		Schnittpunkt mit der Y-Achse; HIRE 1 ··· 3 HIRE 1: N * b HIRE 2: ΣX^2 HIRE 3: $(\Sigma X)^2$
Op 13		Korrelationskoeffizient; HIRE 1 ··· 4 HIRE 1: $\frac{\Sigma Y^2 - \frac{(\Sigma Y)^2}{N}}{a}$ HIRE 2: $\Sigma X^2 - \frac{(\Sigma X)^2}{N}$ HIRE 3: ΣY^2 HIRE 4: $(\Sigma Y)^2$
Op 14		Y für gegebene X-Werte; HIRE 1, 2, 3 und 8 HIRE 1: a HIRE 2: ΣX^2 HIRE 3: $(\Sigma X)^2$ HIRE 8: gegebenes X

Op 15	X für gegebene Y-Werte; HIRE 1, 2, 3 und 8 HIRE 1: (gegebenes Y) − b HIRE 2: ΣX^2 HIRE 3: $(\Sigma X)^2$ HIRE 8: gegebenes Y
Umwandlungsoperationen D.MS	Grad-Min-Sek in Dezimalgrad; HIRE 1, 2 und 8 HIRE 1: $\frac{\text{Gesamtsekunden}}{100}$ HIRE 2: $\frac{\text{Min}}{100} + \frac{\text{Sek}}{10\,000}$ HIRE 8: $\text{Min} + \frac{\text{Sek}}{100}$
INV D.MS	Dezimalgrad in Grad-Min-Sek; HIRE 1, 2 und 8 HIRE 1: $(\text{Grad} * 100) + \text{Min} + \frac{\text{Sek}}{100}$ HIRE 2: $\frac{\text{Sek}}{60}$ HIRE 8: $(\text{Grad} * 100) + \text{Min} + \frac{\text{Sek}}{60}$
P − R	Polar- in rechtwinklige Koordinaten; HIRE 1, 7 und 8 HIRE 1: sin Θ; HIRE 7: R; HIRE 8: Θ
INV P − R	Rechtwinklige in Polarkoordinaten; HIRE 1, 2, 7 und 8 HIRE 1: 0; HIRE 2: Y; HIRE 7: X; HIRE 8: Y

(Kode 82 15), in den Programmspeicher zu bringen ist, dürfte bekannt sein: Man tippt ein

STO 82 STO 15 oder STO 82 E

und löscht den oder die STO-Befehle mit der DEL-Taste.

Problematisch wird es bei der Frage nach dem Befehlskenner. Ein HIR-Befehl hat immer die Form HIR XY.

Dabei steht Y für die Nummer des angesprochenen Registers. Y kann eine Zahl von eins bis acht sein. HIR X0 oder HIR X9-Befehle sind zwar möglich, zeigten aber keine Wirkung (ergaben aber auch keine Fehleranzeige).

Das X steht für den auszuführenden Befehl.

Folgende Befehle sind möglich:

- HIR 0Y Einspeichern eines Wertes in HIRE Y. (STO)
- HIR 1Y Abrufen eines gespeicherten Wertes. (RCL)
- (HIR 2Y keine Wirkung festgestellt.)
- HIR 3Y Addition zu HIRE Y. (SUM)
- HIR 4Y Multiplikation zu HIRE Y. (Prd)
- HIR 5Y Subtraktion vom HIRE-Register Y. (INV SUM)
- HIR 6Y Division von HIRE Y. (INV Prd)
- HIR 7Y bis 9Y wie HIR 6Y.

Als Argumente sind für HIR 0Y und HIR 1Y alle Zahlenwerte zugelassen: positiv, negativ, ganzzahlig, gebrochen, gemischt usw. Die Arithmetik-Befehlsgruppen HIR 3Y bis 9Y (für $Y = 1 \cdots 8$) nehmen keine reinen gebrochenen Zahlen in Kommadarstellung an.

Solche Zahlen, wie 0,6 oder 0,0095, werden von den besagten HIR-Befehlen vor der Ausführung umgewandelt.

Diese Umwandlung erfolgt durch Negation der Zehnerbasispotenz:

Beispiel:

Aus $0{,}6 = 6 * 10^{-1}$ wird $6 * 10^{+1} = 60$

Aus $0{,}0095 = 9{,}5 * 10^{-3}$ wird $9{,}5 * 10^{+3} = 9500$

Mathematisch gesehen wird dabei folgende Funktion angewandt:

$$y = f(x)$$
$$= 10^{(2*\text{Int}(1-\log(|x|)) + \log(|x|))} * \text{Sig}(X)$$

Mit dem so umgewandelten Zahlenwert wird die befohlene Operation durchgeführt. Die Umwandlung unterbleibt aber, wenn die gebrochene Zahl bereits in Exponentialdarstellung vorliegt. Wird also anstelle von 0,6 $6*10^{-1}$ (6 EE −1) eingegeben, so führt der Rechner den Befehl richtig aus.

Die HIR-Befehle sind nicht indirekt zu kennzeichnen. Ein HIR Ind ZZ (Kode 82 40 ZZ) wird immer als HIR 40 interpretiert.

Invertierbar sind nur die HIR 3Y und 4Y-Befehle. So bewirkt ein INV HIR 3Y eine Subtraktion und INV HIR 4Y eine Division, während bei HIR 0Y,1Y und 5Y ··· 9Y die INV oder 2nd INV-Anweisung keine Wirkung hat.

Druckkodes

Die Druckkodes werden bekanntlich mit Op 1 ··· 4 in die HIRE-Register 5 ··· 8 gebracht, dabei aber ebenfalls umgewandelt. So wird aus dem Druckkode für ABCDE, also 1314151617, nach Eingabe mit Op 1 im HIRE-Register 5 die Zahl 0,001314151617. Die drei internen Rundungsstellen werden dabei mit belegt.

Seltsamerweise wird der Druckkode anders interpretiert, wenn er, bereits umgewandelt, mit HIR 0Y in das Register gebracht wurde:

Beispiel:

Eingabe 1314151617 mit Op 1
HIRE-Reg.0,00 1314151617in HIRE 5
Druck A B C D E

Eingabe 0,001314151617mit HIR 05
HIRE-Reg.0,001314151617in HIRE 5
Druck U * %2

Der Rechner wird also veranlaßt, die ersten drei signifikanten Stellen nicht zu beachten und dafür rechts drei Nullen zu lesen. Auch die Stellung des Dezimalpunktes wird nicht beachtet. Der Ausdruck bleibt hier der gleiche, egal ob man 0,001314151617 oder 1314151617. oder 1314.151617 mir HIR eingegeben hat.

Das gilt auch, wenn ein mit Op 1 ··· 4 eingegebener Druckkode mittels HIR-Befehlen arithmetisch bearbeitet wurde (auch wenn sich der Zahlenwert nicht ändert, z.B. Multiplikation mit 1 oder Addition von 0). Lediglich der Abrufbefehl HIR 1Y ändert hier nichts.

Beim Speichern oder Bearbeiten von Druckkodes mit HIR ist also Vorsicht geboten. Man muß dafür Sorge tragen, daß vor dem eigentlichen Druckkode drei Stellen besetzt sind. So ergibt zum Beispiel 1.001314151617 oder 2481314151.617 den richtigen Ausdruck ABCDE. *(bitte umblättern)*

HIR-Register u. -Befehle . . .

Belegung der HIRE-Register
Bei der Benutzung der HIRE-Register als zusätzliche Datenspeicher im Programm muß man darauf achten, daß der Speicherinhalt nicht vom Programm bzw. dem Rechner verändert wird.
Wie bereits bemerkt, werden die HIRE-Register für das Ablegen offener Operationen benutzt. Schon eine einfache Addition belegt HIRE 1 mit der ersten Eingabe. Eine kompliziertere Funktion, wie z.B. $A + B^* C^{(D + E * F)}$ belegt HIRE 1 ··· 5 und kommt damit schon in den Bereich der Druckregister.
Interessant ist dabei allerdings, daß die HIRE-Register auch nach Abschluß der Operation und selbst nach Löschen mit CLR ihren Wert behalten und somit als eine Art Last X-Register verwendbar sind. Im obenstehenden Beispiel sind auch nach Abschluß der Operationen die Werte A ··· E in HIRE 1 ··· 5 unangetastet vorhanden, und können für weitere Berechnungen abgerufen werden.
Aber die HIRE-Register werden vom Rechner auch noch für andere Operationen intern ge-„hired".
Die Liste dieser Operationen siehe vorn.

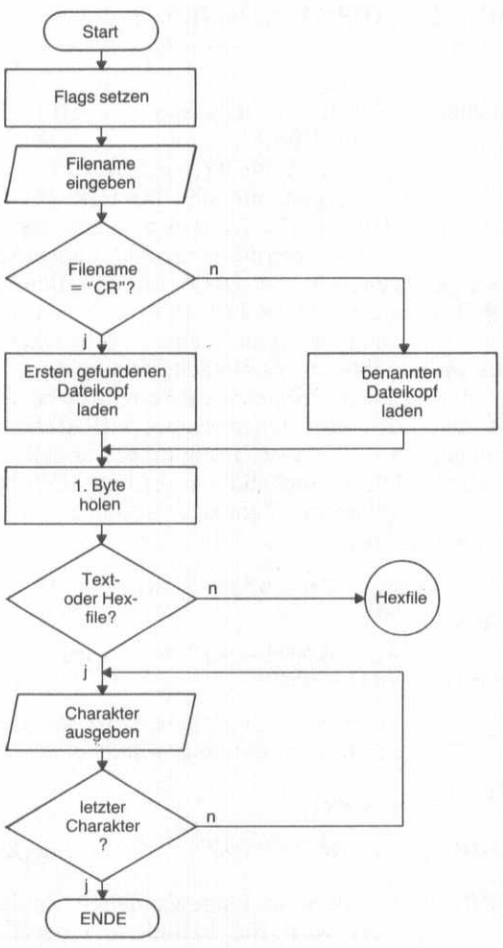
Anmerkung:
Bei den statistischen Funktionen bedeuten:
N — Anzahl der Wertpaare
a — Steigung der Geraden
b — Schnittpunkt mit der Y-Achse
(a und b aus Geradengleichung:
$y = f(X) = aX + b$)
Wie man sieht, gibt es also viele interne „Kunden" für die Mietregister des TI-58/59.
Sollten Sie zufällig einen der oben aufgeführten Werte in Ihrem Programm brauchen, ist das natürlich von Vorteil für Sie.
Ansonsten gilt hier Vorsicht und nochmals Vorsicht, wenn Sie die HIRE-Register als zusätzliche Datenspeicher benutzen möchten (wie oft fehlte bei langen Programmen gerade der 11. oder 21. Speicher!).
Überprüfen Sie schon bei der Programmerstellung genau, welche Befehle Sie benutzen und wieviele Operationen offenstehen, bevor Sie verzweifelt nach der Ursache für Programmfehler suchen müssen.

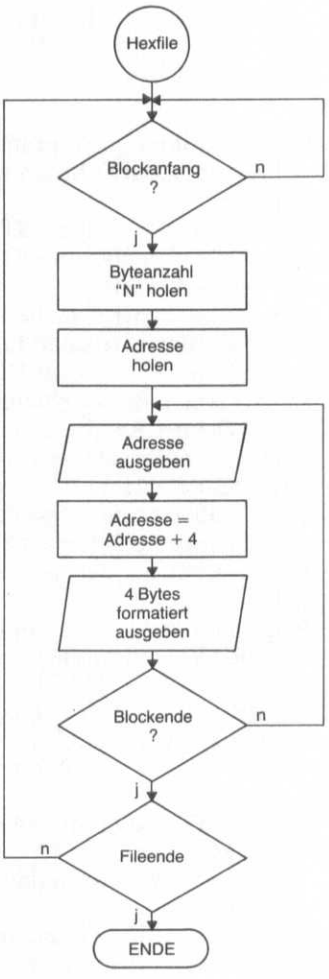
Werner Tratzmüller

Read Tape and Print

Beschreibung

Dieses Programm ist als Dienstprogramm konzipiert.
Gelegentlich möchte man Kassetteninhalte überprüfen, ohne ein Programm oder eine Textdatei unbedingt zu laden. Speziell vom Assembler generierte Objekt-Files sind als Ausdruck interessant, wenn man für irgendwelche Vergleiche, Auswertungen oder Ähnliches den reinen HEX-Maschinenkode benötigt. Ein Objekt-Output auf den AIM-Printer ergibt ein recht unleserliches Format, das voll dem Lochstreifen-Systemformat von Rockwell entspricht. Ein Laden in den Speicher und das entsprechende ausdrucken über einzelne „(M)"-Befehle ist entsprechend mühsam und uneffektiv. Speziell in solchen Fällen leistet das nachstehende Programm nützliche Dienste.
Ein Auslisten von sequentiellen ASCII-Files direkt vom Band ist ebenso möglich, und man kann den Texteditor für andere Anwendungen freihalten.
Das Programm verwendet eine ganze Reihe von Monitor-Subroutinen des „AIM 65". Da aber besagte Routinen mitunter ihre Tücken haben, besonders dann, wenn sie nicht zu den im Handbuch empfohlenen Routinen zählen, ist es teilweise nötig, durch das Umschreiben von Teilstücken und Wahl der richtigen Ansprungziele eine entsprechende Anpassung vorzunehmen.
Da der „AIM"-Monitor es nur zuläßt, TAPE-Files gezielt mittels „Filename" einzulesen, mußte auf die sonst so praktisch zu handhabende Routine „WHE-





RI“ verzichtet werden und eine Ersatzroutine geschrieben werden, die nicht über „FILE-Name“ stolpert.
Zu Beginn des Programmes werden die entsprechenden Flags gesetzt.
Die Subroutine „FNAME“ ist eine Ersatzroutine für die entsprechende Monitorroutine „FNAM“.
Wenn diese Monitorroutine aufgerufen wird und die Rückmeldung in der Anzeige „F=“ mit Space oder Return bestätigt wird, so wird vom Monitor in beiden Fällen ein Filename generiert, der aus 5 SPACE-Charakters besteht.
Die neue Routine dagegen unterscheidet „CR“ und „SPACE“ und kennzeichnet im File-Name-Buffer das erste Byte mit dem Wert „$0D“, der als Unterscheidungskriterium für ein gezieltes mit File-Name b.z.w. ein Laden der logisch nächsten Datei herangezogen wird. So kann der folgende Programmteil, der nur eine modifizierte Version der Monitorroutine „LOADTA“ ist, unterscheiden, ob nach einem File-Name gesucht werden soll oder einfach nur der erste Block einer gefundenen Datei geladen werden soll.
Sobald der File-Haeder gefunden und geladen ist, erfolgt die Entscheidung über die Dateiart und der Ansprung des jeweiligen Programmteils. Die jeweiligen Markierungs-Bytes sind die ASCII-Zeichen „CR“ und „;“.
Bei Objektdateien ist allerdings eine einfache byteweise Leseroutine infolge der Formatverschachtelung von Band- und Lochstreifenformat nicht möglich und man muß über ein entsprechendes Programm dieses doppelte Format berücksichtigen. Eine reine ASCII-Datei dagegen läßt sich völlig unkompliziert auslesen und ausgeben.
Als Hauptanwendung dieses Programmes ist also in erster Linie an nachträgliche Dokumentation vorhandener Dateien gedacht worden. Durch Umschreiben einzelner Programmteile ließen sich sicher noch andere Kassettendienstprogramme verwirklichen.

```
=-0000
;DIESES PROGRAMM WIRD NACH DEM LADEN MIT "F1" AUFGERUFEN !
;WIRD MIT "CR" BESTAETIGT, SO WIRD DIE ERSTE GEFUNDENE DATEI
;GELESEN UND AUSGEDRUCKT.
;BEI EINGABE EINES FILENAMENS WIRD SELEKTIERT GELESEN !
;ASCII-DATEIEN WERDEN OHNE UMWANDLUNG AUSGEDRUCKT.
;MASCHINEN-CODE-DATFIEN WERDEN ALS SPEICHERAUSZUG GEDRUCKT.
;NACH KOMPLETTEM AUSLESEN WIRD DER RECORDER ANGEHALTEN.
;BELIEBIGER NEUSTART MIT "F1" !

=-0000 PRIFLG=$A411
=-0000 CURPO2=$A415
=-0000 ADDR=$A41C
=-0000 CKSUM=$A41E
=-0000 NAME=$A42E
=-0000 TAPIN=$A434
=-0000 TAPTR=$A436
;
=-0000 MONIT=$E182
=-0000 LOADTA=$E32F
=-0000 KEP=$E7AF
=-0000 BLANK2=$E83B
=-0000 BLANK=$E83E
=-0000 WHICHT=$E8A8
=-0000 NAMO3=$E8EB
=-0000 RDRUB=$E95F
=-0000 OUTPUT=$E97A
=-0000 CRLF=$E9F0
=-0000 CRLOW=$EA13
=-0000 NUMA=$EA46
=-0000 PHXY=$EB9E
=-0000 PLXY=$EBAC
=-0000 TIBYTE=$ED3B
=-0000 TIBY1=$ED53
=-0000 OUTDP=$EEFC

;
;
=-0000
        *=$E00
=-0E00
;
;
=-0E00 START
;
203EE8  JSR  BLANK
203BE8  JSR  BLANK2
A900    LDA  #0
8D34A4  STA  TAPIN
A200    LDX  #0
209EEB  JSR  PHXY
=-0E10
20F50E  JSR  FNAME
;
A980    LDA  #$80
8D11A4  STA  PRIFLG
;
AD2EA4  LDA  NAME
F006    BEQ  NXTFL
;
202FE3  JSR  LOADTA

=-0E20
4C340E  JMP  MSG
;
=-0E23 NXTFL
;
A900    LDA  #0
8D1501  STA  $115
2053ED  JSR  TIBY1
CA      DEX
8E15A4  STX  CURPO2
BD1601  LDA  $116,X
D0EF    BNE  NXTFL
=-0E34
;
=-0E34 MSG
;
203BE8  JSR  BLANK2
A946    LDA  #'F'
207AE9  JSR  OUTPUT
A93D    LDA  #'='
207AE9  JSR  OUTPUT
A200    LDX  #0
;
```

(bitte umblättern)

Read Tape and Print

```
==0E43 RLOOP
;
BD1601 LDA $116,X
207AE9 JSR OUTPUT
E8     INX
E006   CPX #6
D0F5   BNE RLOOP
2071E3 JSR $E371
20F0E9 JSR CRLF
==0E54
203BED JSR TIBYTE
C90D   CMP #$0D
F00F   BEQ RDOBJ
20FCEE JSR OUTDP
==0E5E NXTCHR
;
20D90E JSR TBNEU
C900   CMP #0
F069   BEQ FINE
20FCEE JSR OUTDP
D0F4   BNE NXTCHR
;
==0E6A RDOBJ
;
20D90E JSR TBNEU
C93B   CMP #';'
D0F9   BNE RDOBJ
20D90E JSR TBNEU
AA     TAX
20D90E JSR TBNEU
8D1DA4 STA ADDR+1
==0E7B
20D90E JSR TBNEU
8D1CA4 STA ADDR
8A     TXA
F04A   BEQ FINE
2903   AND #3
48     PHA
8A     TXA
4A     LSR A
4A     LSR A
AA     TAX
==0E8B
F03B   BEQ NEXT
;
==0E8D LPRINT
;
A004   LDY #4
;
==0E8F NXT2
;
203EE8 JSR BLANK
AD1DA4 LDA ADDR+1
2046EA JSR NUMA
AD1CA4 LDA ADDR
2046EA JSR NUMA
203BE8 JSR BLANK2
==0EA1
18     CLC
A904   LDA #4
6D1CA4 ADC ADDR
8D1CA4 STA ADDR
9008   BCC NXT1
A900   LDA #0
6D1DA4 ADC ADDR+1
==0EB1
8D1DA4 STA ADDR+1
;
==0EB4 NXT1
;
20D90E JSR TBNEU
2046EA JSR NUMA
203EE8 JSR BLANK
88     DEY
D0F4   BNE NXT1
2013EA JSR CRLOW
CA     DEX
==0EC4
30A4   BMI RDOBJ
D0C5   BNE LPRINT
;
==0EC8 NEXT
;
68     PLA
F09F   BEQ RDOBJ
A8     TAY
D0C1   BNE NXT2
;
==0ECE FINE
;
20F0E9 JSR CRLF
A900   LDA #0
8D11A4 STA PRIFLG
4C82E1 JMP MONIT
;
==0ED9 TBNEU
;
A950   LDA #$50
CD36A4 CMP TAPTR
D011   BNE OLDBLK
209EEB JSR PHXY
AD15A4 LDA $A415
48     PHA
2053ED JSR TIBY1
==0EEA
68     PLA
8D15A4 STA CURPO2
20ACEB JSR PLXY
;
==0EF1 OLDBLK
;
203BED JSR TIBYTE
60     RTS
;
==0EF5 FNAME
;
A04D   LDY #$4D
20AFE7 JSR KEP
A000   LDY #0
;
==0EFC NAM01
;
205FE9 JSR RDRUB
C90D   CMP #$0D
F00A   BEQ NONAME
C920   CMP #' '
D00A   BNE LESE
;
==0F07 REST
;
20EBE8 JSR NAM03
4C190F JMP LESE1
;
==0F0D NONAME
;
A900   LDA #0
F0F6   BEQ REST
;
==0F11 LESE
;
992EA4 STA NAME,Y
C8     INY
C005   CPY #5
D0E3   BNE NAM01
;
==0F19 LESE1
;
203EE8 JSR BLANK
20A8E8 JSR WHICHT
60     RTS
;
;
==0F20
       *=$10C
4C000E JMP START
       .END
```

Christian Meiners

Apple II plus Monitorerweiterung

Beschreibung

Für den APPLE gibt es 2 unterschiedliche Grundausführungen:

- APPLE II mit INTEGER-BASIC und SYSTEM-MONITOR. Der System-Monitor enthält die Befehle S (Step) und T (Trace) zum schrittweisen Abarbeiten von Maschinenprogrammen. Im Integer-Basic-ROM ist der Miniassembler abgelegt.
- APPLE II plus mit APPLESOFT-BASIC und AUTOSTART-MONITOR. Hier wurde auf die S- bzw. T-Befehle zugunsten der Selbststarteigenschaften des Systems verzichtet. Auch der Miniassembler fehlt.

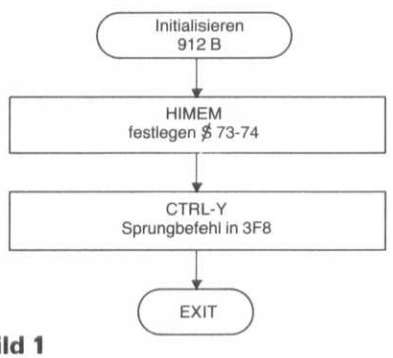


Bild 1

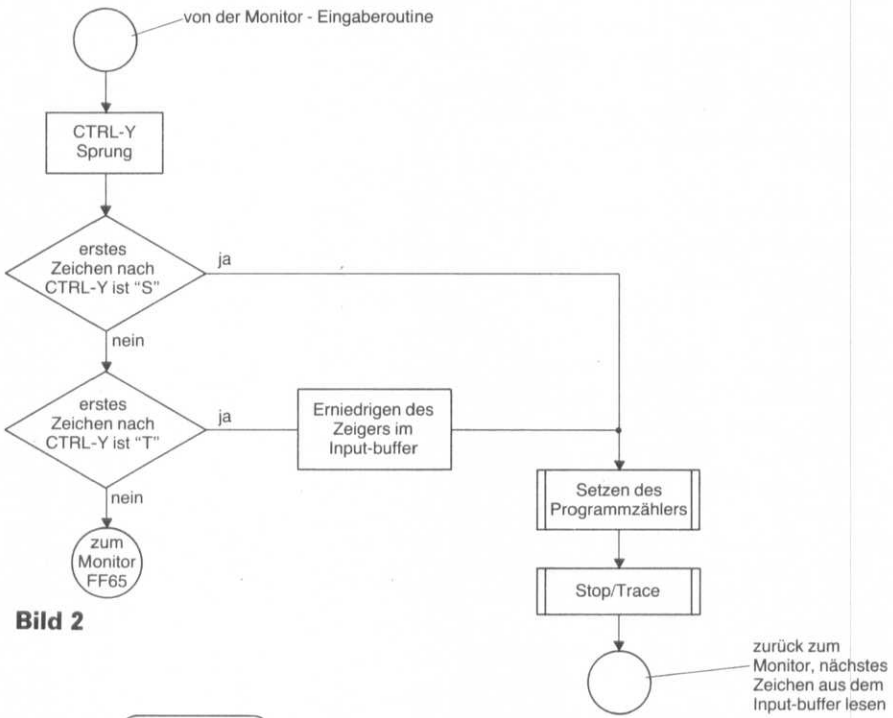


Bild 2

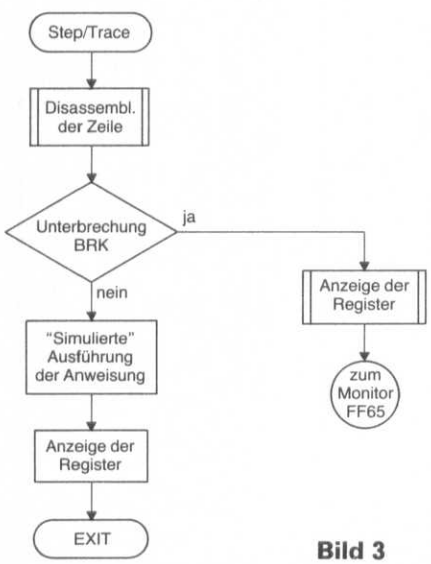


Bild 3

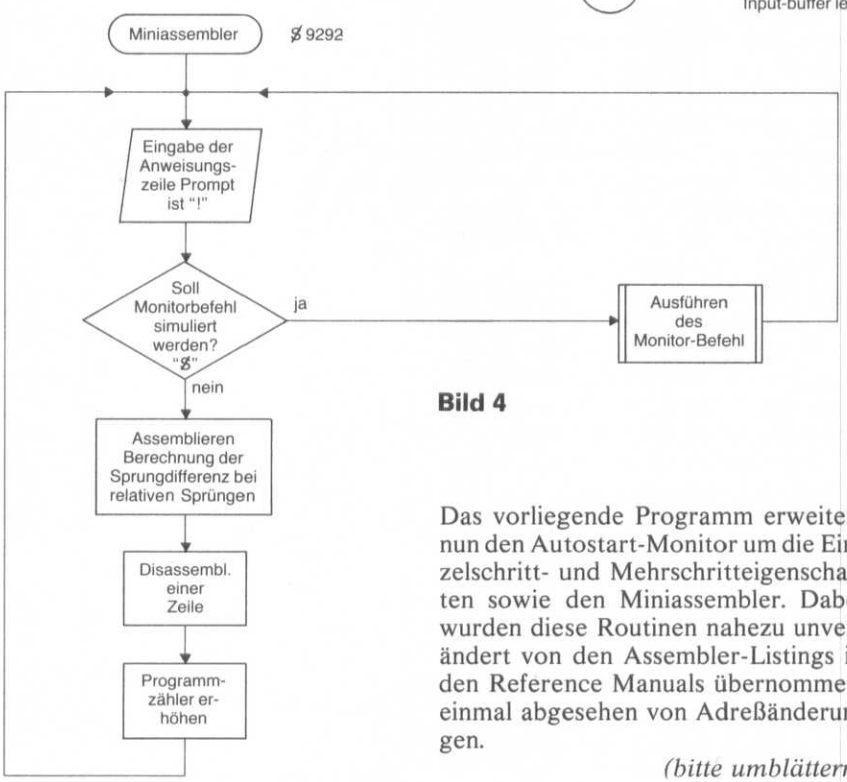


Bild 4

Das vorliegende Programm erweitert nun den Autostart-Monitor um die Einzelschritt- und Mehrschritteigenschaften sowie den Miniassembler. Dabei wurden diese Routinen nahezu unverändert von den Assembler-Listings in den Reference Manuals übernommen, einmal abgesehen von Adreßänderungen.

(bitte umblättern)

Apple II plus Monitorerweiterung

Das Programm läßt sich in 3 Routinen einteilen:
- Initialisierungsprogramm:
- Step/Trace Programm
- Miniassembler

Initialisierungsprogramm:

Adreßbereich (alle Zahlen hexadezimal): 912B...9142. Das Programm hat die Aufgabe, die Routinen gegen Überschreiben durch Basic-Programme/-Variable zu schützen (HIMEM).
Weiter wird der CTRL-Y-Sprungbefehl in 3F8 für das Step/Trace-Programm eingerichtet (Bild 1).

Einzelschritt- und Mehrschrittbefehle (Step/Trace):

Adreßbereich: 9143...91F1 und 9343...9366.
Der Autostart-Monitor enthält die Möglichkeit, mit Hilfe des CTRL-Y Zeichens benutzerspezifische Befehle zu definieren. Diese Eigenschaft wird ausgenutzt indem man vor „S" bzw. „T" das Zeichen CTRL-Y schreibt. Dieses Kontrollzeichen ist nicht auf dem Bildschirm sichtbar. Zur besseren Übersicht wird es im folgenden in Klammern gesetzt: (CTRL-Y) (Bild 2).
Findet nun die Monitor-Eingaberoutine das Zeichen (CTRL-Y), so führt der Monitor einen Sprung zur Adresse 3F8 durch. Hier steht der Sprungbefehl in das eigentliche Programm.
3F8 — JMP $91DA
Dem Step/Trace-Programm ist ein Hilfsprogramm vorangestellt, welches bei unzulässigen Eingaben wieder zum Monitor zurückkehrt bzw. entsprechende Sprünge für (CTRL-Y)S und (CTRL-Y)T vornimmt.
Beim Mehrschrittbefehl (CTRL-Y)T wird eigentlich in einer Endlosschleife ein Einzelschritt immer wieder ausgeführt. Dies erreicht man durch Manipulation des Zeigers $34. Dieser Zeiger weist auf das nächste zu lesende Byte im Input-buffer. Wird dieser Zeiger nach jedem Lesevorgang wieder erniedrigt, so wird immer wieder der gleiche Befehl gelesen und ausgeführt (Bild 3).
Die Step/Trace-Routine disassembliert zunächst die auszuführende Anweisung, bei einer Unterbrechung (BRK) wird dann das Programm beendet, sonst erfolgt eine simulierte Ausführung der Anweisung und danach die Anzeige der Register.

```
912B-  A9 2A     LDA  #$2A
912D-  85 73     STA  $73
912F-  A9 4C     LDA  #$4C
9131-  8D 78 03  STA  $0378
9134-  A9 DA     LDA  #$DA
9136-  8D F9 03  STA  $03F9
9139-  A9 91     LDA  #$91
913B-  8D FA 03  STA  $03FA
913E-  85 74     STA  $74
9140-  4C 03 E0  JMP  $E003
9143-  20 D0 FB  JSR  $FBD0
9146-  68        PLA
9147-  85 2C     STA  $2C
9149-  68        PLA
914A-  85 2D     STA  $2D
914C-  A2 08     LDX  #$08
914E-  BD 56 93  LDA  $9356,X
9151-  95 3C     STA  $3C,X
9153-  CA        DEX
9154-  D0 F8     BNE  $914E
9156-  A1 3A     LDA  ($3A,X)
9158-  F0 42     BEQ  $919C
915A-  A4 2F     LDY  $2F
915C-  C9 20     CMP  #$20
915E-  F0 59     BEQ  $91B9
9160-  C9 60     CMP  #$60
9162-  F0 45     BEQ  $91A9
9164-  C9 4C     CMP  #$4C
9166-  F0 5C     BEQ  $91C4
9168-  C9 6C     CMP  #$6C
916A-  F0 59     BEQ  $91C5
916C-  C9 40     CMP  #$40
916E-  F0 35     BEQ  $91A5
9170-  29 1F     AND  #$1F
9172-  49 14     EOR  #$14
9174-  C9 04     CMP  #$04
9176-  F0 02     BEQ  $917A
9178-  B1 3A     LDA  ($3A),Y
917A-  99 3C 00  STA  $003C,Y
917D-  88        DEY
917E-  10 F8     BPL  $9178
9180-  20 3F FF  JSR  $FF3F
9183-  4C 3C 00  JMP  $003C
9186-  85 45     STA  $45
9188-  68        PLA
9189-  48        PHA
918A-  0A        ASL
918B-  0A        ASL
918C-  0A        ASL
918D-  30 03     BMI  $9192
918F-  6C FE 03  JMP  ($03FE)
9192-  28        PLP
9193-  20 4C FF  JSR  $FF4C
9196-  68        PLA
9197-  85 3A     STA  $3A
9199-  68        PLA
919A-  85 3B     STA  $3B
919C-  20 B2 FB  JSR  $FBB2
919F-  20 DA FA  JSR  $FADA
91A2-  4C 69 FF  JMP  $FF69
91A5-  18        CLC
91A6-  68        PLA
91A7-  85 48     STA  $48
91A9-  68        PLA
91AA-  85 3A     STA  $3A
91AC-  68        PLA
91AD-  85 3B     STA  $3B
91AF-  A5 2F     LDA  $2F
91B1-  20 56 F9  JSR  $F956
91B4-  84 3B     STY  $3B
91B6-  18        CLC
91B7-  90 14     BCC  $91CD
91B9-  18        CLC
91BA-  20 54 F9  JSR  $F954
91BD-  AA        TAX
91BE-  98        TYA
91BF-  48        PHA
91C0-  8A        TXA
91C1-  48        PHA
91C2-  A0 02     LDY  #$02
91C4-  18        CLC
91C5-  B1 3A     LDA  ($3A),Y
91C7-  AA        TAX
91C8-  88        DEY
91C9-  B1 3A     LDA  ($3A),Y
91CB-  86 3B     STX  $3B
91CD-  85 3A     STA  $3A
91CF-  B0 F3     BCS  $91C4
91D1-  A5 2D     LDA  $2D
91D3-  48        PHA
91D4-  A5 2C     LDA  $2C
91D6-  48        PHA
91D7-  4C D7 FA  JMP  $FAD7
91DA-  A4 34     LDY  $34
91DC-  B9 00 02  LDA  $0200,Y
91DF-  C9 D3     CMP  #$D3
91E1-  F0 09     BEQ  $91EC
91E3-  C9 D4     CMP  #$D4
91E5-  F0 03     BEQ  $91EA
91E7-  4C 65 FF  JMP  $FF65
91EA-  C6 34     DEC  $34
91EC-  20 75 FE  JSR  $FE75
91EF-  4C 43 91  JMP  $9143
91F2-  00        BRK
91F3-  00        BRK
91F4-  00        BRK
91F5-  00        BRK
91F6-  00        BRK
91F7-  00        BRK
91F8-  00        BRK
91F9-  00        BRK
91FA-  4C 92 92  JMP  $9292
91FD-  00        BRK
91FE-  00        BRK
91FF-  00        BRK
9200-  E9 81     SBC  #$81
9202-  4A        LSR
9203-  D0 14     BNE  $9219
9205-  A4 3F     LDY  $3F
9207-  A6 3E     LDX  $3E
9209-  D0 01     BNE  $920C
920B-  88        DEY
920C-  CA        DEX
920D-  8A        TXA
920E-  18        CLC
920F-  E5 3A     SBC  $3A
9211-  85 3E     STA  $3E
9213-  10 01     BPL  $9216
9215-  C8        INY
9216-  98        TYA
9217-  E5 3B     SBC  $3B
9219-  D0 6B     BNE  $9286
921B-  A4 2F     LDY  $2F
921D-  B9 3D 00  LDA  $003D,Y
9220-  91 3A     STA  ($3A),Y
9222-  88        DEY
9223-  10 F8     BPL  $921D
9225-  20 1A FC  JSR  $FC1A
9228-  20 1A FC  JSR  $FC1A
922B-  20 D0 F8  JSR  $F8D0
922E-  20 53 F9  JSR  $F953
9231-  84 3B     STY  $3B
9233-  85 3A     STA  $3A
9235-  4C 95 92  JMP  $9295
9238-  20 BE FF  JSR  $FFBE
923B-  A4 34     LDY  $34
923D-  20 A7 FF  JSR  $FFA7
9240-  84 34     STY  $34
9242-  A0 17     LDY  #$17
```

```
9244-  88          DEY
9245-  30 4B       BMI   $9292
9247-  D9 CC FF    CMP   $FFCC,Y
924A-  D0 F8       BNE   $9244
924C-  C0 15       CPY   #$15
924E-  D0 E8       BNE   $9238
9250-  A5 31       LDA   $31
9252-  A0 00       LDY   #$00
9254-  C6 34       DEC   $34
9256-  20 00 FE    JSR   $FE00
9259-  4C 95 92    JMP   $9295
925C-  A5 3D       LDA   $3D
925E-  20 BE F8    JSR   $F8BE
9261-  AA          TAX
9262-  BD 00 FA    LDA   $FA00,X
9265-  C5 42       CMP   $42
9267-  D0 13       BNE   $927C
9269-  BD C0 F9    LDA   $F9C0,X
926C-  C5 43       CMP   $43
926E-  D0 0C       BNE   $927C
9270-  A5 44       LDA   $44
9272-  A4 2E       LDY   $2E
9274-  C0 9D       CPY   #$9D
9276-  F0 88       BEQ   $9200
9278-  C5 2E       CMP   $2E
927A-  F0 9F       BEQ   $921B
927C-  C6 3D       DEC   $3D
927E-  D0 DC       BNE   $925C
9280-  E6 44       INC   $44
9282-  C6 35       DEC   $35
9284-  F0 D6       BEQ   $925C
9286-  A4 34       LDY   $34
9288-  98          TYA
9289-  AA          TAX
928A-  20 4A F9    JSR   $F94A
928D-  A9 DE       LDA   #$DE
928F-  20 ED FD    JSR   $FDED
9292-  20 3A FF    JSR   $FF3A
9295-  A9 A1       LDA   #$A1
9297-  85 33       STA   $33
9299-  20 67 FD    JSR   $FD67
929C-  20 C7 FF    JSR   $FFC7
929F-  AD 00 02    LDA   $0200
92A2-  C9 A0       CMP   #$A0
92A4-  F0 13       BEQ   $92B9
92A6-  C8          INY
92A7-  C9 A4       CMP   #$A4
92A9-  F0 92       BEQ   $923D
92AB-  88          DEY
92AC-  20 A7 FF    JSR   $FFA7
92AF-  C9 93       CMP   #$93
92B1-  D0 D5       BNE   $9288
92B3-  8A          TXA
92B4-  F0 D2       BEQ   $9288
92B6-  20 7B FE    JSR   $FE7B
92B9-  A9 03       LDA   #$03
92BB-  85 3D       STA   $3D
92BD-  20 34 93    JSR   $9334
92C0-  0A          ASL
92C1-  E9 BE       SBC   #$BE
92C3-  C9 C2       CMP   #$C2
92C5-  90 C1       BCC   $9288
92C7-  0A          ASL
92C8-  0A          ASL
92C9-  A2 04       LDX   #$04
92CB-  0A          ASL
92CC-  26 42       ROL   $42
92CE-  26 43       ROL   $43
92D0-  CA          DEX
92D1-  10 F8       BPL   $92CB
92D3-  C6 3D       DEC   $3D
92D5-  F0 F4       BEQ   $92CB
92D7-  10 E4       BPL   $928D
92D9-  A2 05       LDX   #$05
92DB-  20 34 93    JSR   $9334
92DE-  84 34       STY   $34
92E0-  DD B4 F9    CMP   $F9B4,X
92E3-  D0 13       BNE   $92F8
92E5-  20 34 93    JSR   $9334
92E8-  DD BA F9    CMP   $F9BA,X
92EB-  F0 0D       BEQ   $92FA
92ED-  BD BA F9    LDA   $F9BA,X
92F0-  F0 07       BEQ   $92F9
92F2-  C9 A4       CMP   #$A4
92F4-  F0 03       BEQ   $92F9
92F6-  A4 34       LDY   $34
92F8-  18          CLC
92F9-  88          DEY
92FA-  26 44       ROL   $44
92FC-  E0 03       CPX   #$03
92FE-  D0 0D       BNE   $930D
9300-  20 A7 FF    JSR   $FFA7
9303-  A5 3F       LDA   $3F
9305-  F0 01       BEQ   $9308
9307-  E8          INX
9308-  86 35       STX   $35
930A-  A2 03       LDX   #$03
930C-  88          DEY
930D-  86 3D       STX   $3D
930F-  CA          DEX
9310-  10 C9       BPL   $92DB
9312-  A5 44       LDA   $44
9314-  0A          ASL
9315-  0A          ASL
9316-  05 35       ORA   $35
9318-  C9 20       CMP   #$20
931A-  B0 06       BCS   $9322
931C-  A6 35       LDX   $35
931E-  F0 02       BEQ   $9322
9320-  09 80       ORA   #$80
9322-  85 44       STA   $44
9324-  84 34       STY   $34
9326-  B9 00 02    LDA   $0200,Y
9329-  C9 8B       CMP   #$8B
932B-  F0 04       BEQ   $9331
932D-  C9 8D       CMP   #$8D
932F-  D0 80       BNE   $92B1
9331-  4C 5C 92    JMP   $925C
9334-  B9 00 02    LDA   $0200,Y
9337-  C8          INY
9338-  C9 A0       CMP   #$A0
933A-  F0 F8       BEQ   $9334
933C-  60          RTS
933D-  00          BRK
933E-  00          BRK
933F-  00          BRK
9340-  4C 92 92    JMP   $9292
9343-  18          CLC
9344-  A0 01       LDY   #$01
9346-  B1 3A       LDA   ($3A),Y
9348-  20 56 F9    JSR   $F956
934B-  85 3A       STA   $3A
934D-  98          TYA
934E-  38          SEC
934F-  B0 0F       BCS   $9360
9351-  20 4A FF    JSR   $FF4A
9354-  38          SEC
9355-  B0 0C       BCS   $9363
9357-  EA          NOP
9358-  EA          NOP
9359-  4C 51 93    JMP   $9351
935C-  4C 43 93    JMP   $9343
935F-  00          BRK
9360-  4C AD 91    JMP   $91AD
9363-  4C AF 91    JMP   $91AF
9366-  00          BRK
9367-  00          BRK
```

Die zu testende Programmanweisung wird dabei in einen Ausführungsbereich auf der 0. Seite übertragen und dort ausgeführt.
Der erste Einzelschritt- bzw. Mehrschrittbefehl muß die Anfangsadresse des zu prüfenden Programms enthalten, z.B.:
300(CTRL-Y)S
Weitere Schritte werden mit jeweils: (CTRL-Y)S ausgeführt.
Da der (CTRL-Y)T Befehl eine Endlosschleife erzeugt, welche nur durch BRK zu stoppen ist, sollte die letzte Programmanweisung ein BRK sein. Es kann sonst passieren, daß sich das Step/Trace-Programm aufhängt, d.h. unkontrollierbar wird oder aber endlos läuft. In diesen Fällen hilft dann nur noch die Notbremse RESET.
Die Stop/List-Einrichtung des Autostart-Monitors ermöglicht eine kontrollierte Mehrschrittlistung mit Zwischenstops.

Miniassembler:

Adreßbereich: 91FA...933D.
Assembler erleichtern die Programmierung von Maschinenprogrammen. Für kürzere, übersichtliche Programme reicht ein sog. Miniassembler aus. Er kann im Gegensatz zum Makroassembler keine Label oder Namen verstehen. Auch fehlt die Möglichkeit, erläuternde Kommentare in das Programm zu schreiben.
Sprungadressen, Speicherzellen und Hex-Werte müssen als Hex-Zahlen eingegeben werden. Nur die eigentliche Instruktion (Mnemonie-code) wird assembliert. Beim Miniassembler wird jede Eingabezeile sofort in Maschinenkode umgesetzt. Programmzählerdifferenzen für relative Sprünge werden errechnet.
Der Miniassembler kann wie folgt aufgerufen werden:
CALL 37370 vom BASIC
9292 G vom Monitor.
Es erscheint dann als Prompt ein Ausrufezeichen „!". Die Anfangsadresse für die erste Programmzeile muß als Hexzahl eingegeben werden. Dann folgt ohne Leerstelle ein Doppelpunkt und dann die Anweisung:
300: LDA $2A
$-Zeichen zur Kennzeichnung einer Hexzahl können weggelassen werden:
300:LDA 2A
Sollen Hexziffern als Zahlenwerte und nicht als Adressen interpretiert wer-

(bitte umblättern)

Apple II plus Monitorerweiterung

den, so muß ein Zeichen vor die betreffende Hexzahl geschrieben werden.
LDA #2A
Auf vorangegangene Zeilen folgende Anweisungen müssen eine Leerstelle als erstes Zeichen haben:
_LDA #2A
Fehlerhafte Eingaben werden vom Rechner mit einem Piepton und einer Markierung „^" der Fehlerstelle quittiert.
LDA 2A
^
Vom Miniassembler lassen sich alle Monitorbefehle ausführen. Erstes Zeichen der Eingabezeile ist dann ein $, gefolgt vom Monitorbefehl.
Beispielsweise kann man ein zuvor mit dem Miniassembler erstelltes Programm mit $...G ausführen oder mit $...(CTRL-Y)S schrittweise abarbeiten, ohne die Programmierebene des Miniassemblers zu verlassen. Ein Rücksprung in den Monitor (Prompt: „*") ist mit $FF65G möglich.

System-Konfiguration und Disk II:

Zur Erstellung der Programme wurde ein APPLE II plus mit 48-kB-RAM und DISK II (DOS 3.3) benutzt. Die Routinen belegen den Adreßbereich unterhalb des DOS (zulässig: MAXFILES kleiner als 5).
Das Initialisierungsprogramm liegt am Anfang. Die Routinen können also mit BRUN von Diskette geladen und sofort initialisiert werden. Sog. Turnkey-Disketten, die diesen Vorgang beim Einschalten des APPLE automatisch erledigen, sind ebenfalls möglich.
APPLE II plus-Benutzer, welche mit Kassettenrecorder arbeiten, können mit den Befehlen:

912B.9366 R von Kassette lesen

bzw. 912B.9366 W auf Kassette schreiben.
Nach dem Einlesen muß mit
912B G initialisiert werden.

Hinweise zur Anpassung an kleinere Systeme

Jeder APPLE II plus-Benutzer weiß sicher selbst am besten, wo er die 570 Byte Maschinenprogramm unterbringt. Die Anpassung ist einfach zu gestalten, wenn das Programm im RAM-Bereich seitenweise (d.h. um je 256 Byte) verschoben wird.
Dann ändern sich folgende Speicherinhalte (Adressen sind natürlich ebenfalls verschoben):

913A : 91	92DD : 93
9150 : 93	9333 : 92
91F1 : 91	9242 : 92
91FC : 92	935B : 93
9237 : 92	935E : 93
925B : 92	9365 : 91
92BF : 93	9362 : 91
92E7 : 93	

Beispiel: Verschieben um −30 Seiten:
aus 9365 : 91 wird 6365 : 61
Die oben beschriebenen Routinen verbinden die Möglichkeiten und Vorteile von AUTOSTART- und SYSTEM-MONITOR.
Zum Erstellen von kürzeren Maschinenprogrammen reicht der Miniassembler aus und die Einzelschritt- bzw. Mehrschrittbefehle erleichtern die Fehlersuche in Maschinenprogrammen.

Literatur:

[1] APPLE II Reference Manual.
[2] APPLE II Reference Manual, January 1978.
[3] Peter Heuer, Microcomputer-Programmierung

Bildschirmmeldung einer Programmdiskette

```
MASTER DISK  PROGRAMM-ERZEUGUNG (V254)

STEP/TRACE UND MINI-ASSEMBLER
OBERHALB HIMEM INITIALISIERT:

   STEP  = (CTRL-Y)S
   TRACE = (CTRL-Y)T

AUFRUF DES MINI-ASSEMBLERS:

   VON BASIC : CALL 37370
   VOM MONITOR : 9292G

HILFSPROGRAMME :

   RUN  RENUMBER          (APPLESOFT)
   BRUN COMPRESS          (APPLESOFT)
   BRUN INTEGER           (INTEGER-BASIC)
```

```
9120- 00 04 00 A9 2A 85 73 A9
9130- 4C 8D 78 03 A9 DA 8D F9
913B- 03 A9 91 8D FA 03 85 74
9140- 4C 03 E0 20 D0 F8 68 85
9148- 2C 68 85 2D A2 08 BD 56
9150- 93 95 3C CA D0 F8 A1 3A
9158- F0 42 A4 2F C9 20 F0 59
9160- C9 60 F0 45 C9 4C F0 5C
9168- C9 6C F0 59 C9 40 F0 35
9170- 29 1F 49 14 C9 04 F0 02
9178- B1 3A 99 3C 00 88 10 F8
9180- 20 3F FF 4C 3C 00 85 45
9188- 68 48 0A 0A 0A 30 03 6C
9190- FE 03 28 20 4C FF 68 85
9198- 3A 68 85 3B 20 82 F8 20
91A0- DA FA 4C 65 FF 18 68 F5
91A8- 48 68 85 3A 68 85 3B A5
91B0- 2F 20 56 F9 84 3B 1F 90
91B8- 14 18 20 54 F9 AA 98 48
91C0- BA 48 A0 02 18 B1 3A AA
91C8- 88 B1 3A 86 3B 85 3A B0
91D0- F3 A5 2D 48 A5 2C 48 4C
91D8- D7 FA A4 34 B9 00 02 C9
91E0- D3 F0 09 C9 D4 F0 03 4C
91E8- 65 FF C6 34 20 75 FE 4C
91F0- 43 91 00 00 00 00 00 00
91F8- 00 00 4C 92 92 00 00 00
9200- E9 81 4A D0 14 A4 3F A6
9208- 3E D0 01 88 CA 8A 18 E5
9210- 3A 85 3E 10 01 C8 98 E5
9218- 3B D0 68 A4 2F B9 3D 00
9220- 91 3A 88 10 F8 20 1A FC
9228- 20 1A FC 20 D0 F8 20 53
9230- F9 84 3B 85 3A 4C 95 92
9238- 20 BE FF A4 34 20 A7 FF
9240- 84 34 A0 17 88 30 4B D9
9248- CC FF D0 F8 C0 15 D0 F8
9250- A5 31 A0 00 C6 34 20 00
9258- FE 4C 95 92 A5 3D 20 8E
9260- F8 AA BD 00 FA C5 42 D0
9268- 13 BD C0 F9 C5 43 D0 0C
9270- A5 44 A4 2E C0 9D F0 08
9278- C5 2E F0 9F C6 3D D0 DC
9280- E6 44 C6 35 F0 D6 A4 34
9288- 98 AA 20 4A F9 A9 DE 20
9290- ED FD 20 3A FF A9 A1 85
9298- 33 20 67 FD 20 C7 FF AD
92A0- 00 02 C9 A0 F0 13 C8 C9
92A8- A4 F0 92 88 20 A7 FF C9
92B0- 93 D0 D5 8A F0 D2 20 78
92B8- FE A9 03 85 3D 20 34 93
92C0- 0A E9 BE C9 C2 90 C1 0A
92C8- 0A A2 04 0A 26 42 26 43
92D0- CA 10 F8 C6 3D F0 F4 10
92D8- E4 A2 05 20 34 93 84 34
92E0- DD B4 F9 D0 13 20 34 93
92E8- DD BA F9 F0 0D BD BA F9
92F0- F0 07 C9 A4 F0 03 A4 34
92F8- 18 88 26 44 E0 03 D0 0D
9300- 20 A7 FF A5 3F F0 01 E8
9308- 86 35 A2 03 88 86 3D CA
9310- 10 C9 A5 44 0A 0A 85 35
9318- C9 20 B0 06 A6 35 F0 02
9320- 09 80 85 44 84 34 B9 00
9328- 02 C9 88 F0 04 C9 8D D0
9330- 80 4C 5C 92 B9 00 02 C8
9338- C9 A0 F0 F8 60 00 00 00
9340- 4C 92 92 18 A0 01 B1 3A
9348- 20 56 F9 85 3A 98 38 B0
9350- 0F 20 4A FF 38 B0 0C EA
9358- EA 4C 51 93 4C 43 93 00
9360- 4C AD 91 4C AF 91 00 00
```

Heino Velder

Trace für CBM 2001

Beschreibung:

Maschinenprogramme sind schnell und meist auch kürzer als vergleichbare BASIC-Programme. Leider kommt es aber oft vor, daß der CBM wegen eines Fehlers das Programm verläßt und sich in einer Schleife festhängt. Hier hilft dann nur noch ein RESET (oder schalten Sie den CBM immer noch ab?).

Mit dem TRACE-Programm können Sie nun auch Maschinenprogramm schrittweise verfolgen und jederzeit abbrechen. Während das Testprogramm langsam abläuft, werden in den ersten beiden Zeilen die wichtigsten Angaben ausgedruckt (siehe Anfang des Assemblerlistings).

Besonders hervorzuheben ist die Anzeige der Flags, so daß Sie mit Hilfe des TRACE-Programms und des Assemblerlistings genau verfolgen können, ob die gewünschten Verzweigungen oder arithmetischen Operationen durchgeführt werden.

Das Prinzip, nach dem TRACE arbeitet, ist denkbar einfach. Beim CBM wird durch einen Interrupt alle $^1/_{60}$ s eine Routine durchlaufen, um die Tastatur abzufragen. Danach kehrt der Computer wieder zum Hauptprogramm zurück. Man kann nun die Interruptroutine auf fast $^1/_{60}$ s verlagern, so daß, kurz nach dem Rücksprung ins Hauptprogramm, der nächste Interrupt erfolgt. Sicherlich erfordert der genaue Abgleich sehr viel Geschicklichkeit, aber zuletzt bleibt dem Prozessor zwischen den einzelnen Interruptimpulsen nur die Zeit, um lediglich einen Befehl im Hauptprogramm auszuführen.

Wie man gleich erkennt, steht und fällt das Programm mit dem genauen Abgleich der Interruptroutine. Auf das Programm soll hier nicht im einzelnen eingegangen werden, da es weitestgehend durch die Anmerkungen im Assemblerlisting erklärt wird. Sicherlich wird dem einen oder anderen Leser mancher Teil etwas umständlich erscheinen. Sie sollten aber nicht vergessen, daß das Programm unter allen Umständen die gleiche Laufzeit erzielen muß; unabhängig vom jeweiligen Stand der Anzeige. *(bitte umblättern)*

```
TRACE bei Maschinenprogrammen


Es ergibt sich folgende Anzeige:

 PC  AC XR YR SP   C Z   D B I V N
033A 00 05 B4 F2   - ↑ - - ↑ - - ↑

PC: Programmzähler                  C: Carry     Flag
AC: Accumulator                     Z: Zero       "
XR: Indexregister X                 D: Dezimal    "
YR  Indexregister Y                 B: Break      "
SP  Stapelzeiger                    I: Interrupt  "
                                    V: Overflow   "
                                    N: Negativ    "

17A0:   78          SEI             ┐
17A1:   AD 19 02    LDA   $0219     |
17A4:   49 30       EOR   =$30      | IRQ-Vektor
17A6:   8D 19 02    STA   $0219     |
17A9:   AD 1A 02    LDA   $021A     | auf $17B5
17AC:   49 F1       EOR   =$F1      |
17AE:   8D 1A 02    STA   $021A     |
17B1:   58          CLI             ┘
17B2:   4C 57 19    JMP   $1957     := zurück in Monitor
17B5:   A2 FA       LDX   =$FA      ┐
17B7:   68          PLA             | Stack-Daten holen
17B8:   9D FE 02    STA   $02FE,X   |
17BB:   E8          INX             | und abspeichern
17BC:   D0 F9       BNE   $17B7     ┘
17BE:   AD 12 E8    LDA   $E812     ← PIA für Tastatur
17C1:   C9 BF       CMP   =$BF      ← Prüft auf '.'
17C3:   D0 0C       BNE   $17D1     :
17C5:   A5 01       LDA   $01       ┐= $01 low-Byte und
17C7:   8D FC 03    STA   $03FC     |
17CA:   A5 02       LDA   $02       |= $02 high-Byte für
17CC:   8D FD 03    STA   $03FD     |  die Start-Adresse
17CF:   D0 0D       BNE   $17DE     ┘
17D1:   A0 01       LDY   =$01      ┐
17D3:   88          DEY             |
17D4:   D0 FD       BNE   $17D3     | Verzögerung
17D6:   F0 02       BEQ   $17DA     |
17D8:   EA          NOP             | gleicht die beiden
17D9:   EA          NOP             | oberen Wege aus
17DA:   EA          NOP             |
17DB:   A5 EA       LDA   $EA       |
17DD:   EA          NOP             ┘
17DE:   A2 06       LDX   =$06      ┐
17E0:   BD F7 03    LDA   $03F7,X   | Daten zurück
17E3:   48          PHA             | auf den Stack
17E4:   CA          DEX             |
17E5:   D0 F9       BNE   $17E0     ┘
17E7:   BA        TSXLDX            ┐ Zeiger in Stack,
17E8:   8E FE 03    STX   $03FE     ┘ und abspeichern
17EB:   A2 26       LDX   =$26      ┐
17ED:   BD D9 18    LDA   $18D9,X   | Schreibt die
17F0:   9D FF 7F    STA   $7FFF,X   |
17F3:   CA          DEX             | Registeranzeige
17F4:   D0 F7       BNE   $17ED     ┘
17F6:   AD FC 03    LDA   $03FC     ┐
17F9:   20 B0 18    JSR   $18B0     |
17FC:   8E 2B 80    STX   $802B     | Schreibt:
17FF:   8D 2C 80    STA   $802C     |
1802:   AD FD 03    LDA   $03FD     | Programmzähler
```

(Fortsetzung s. nächste Seite)

Trace für CBM 2001

Zu den Befehlen:
Die unterste Tastenreihe des CBM dient der Eingabe. Hat man das Programm durch einen Monitor mit G 17A0 (Go 17A0) aufgerufen, erscheinen sofort die Kontrollzeilen am oberen Bildschirmrand. Da hiermit jedoch das Programm lediglich aufgerufen wird, muß sich in Zeile $17B2 ein korrekter Rücksprung in den Monitor anschließen. Diese Adresse (wie alle mit „=“ gekennzeichneten Stellen) müssen Sie gegebenenfalls ändern. Wenn Sie die Zeile 17B2 in — EA EA 60 — abändern, kann das Programm mit SYS 6048 aufgerufen werden.
Mit der REVERSE-Taste wird dann auf TRACE-Betrieb umgeschaltet, oder mit der „C“-Taste (zwischen SPACE und REVERSE) wieder abgeschaltet. Haben Sie die REVERSE-Taste gedrückt, dann sollte die Anzeige sich nicht mehr verändern. Nun können Sie durch Drücken der „=“-Taste das Programm langsam ablaufen lassen.
Natürlich soll ein Maschinenprogramm von einer definierten Stelle aus durchgetestet werden. Hierzu gibt man die Startadresse (low u. high Byte) in die Speicherzellen, $01, 02 ein. Dann drükken Sie die SPACE-Taste, bis der Programmzähler diese Adresse anzeigt. Drücken Sie nun die „=“-Taste, und Sie können verfolgen, wie Ihr Programm abgearbeitet wird. Durch erneutes Drücken der SPACE-Taste können Sie jederzeit zum Anfang Ihres Testprogramms zurückkehren.
Wenn der Test beendet ist oder der Fehler gefunden wurde, gelangen Sie durch die STOP-Taste wieder in dem Monitor. Zuvor müssen Sie aber, wie zu Beginn, in den Zeilen $189A, 189F die korrekte Rücksprungadresse angegeben haben. Mit 189A: A9 89 und 189F: A9 C3 erfolgt ein Sprung ins Betriebssystem (BASIC-Warm-Start: $C389).
Damit Ihr Monitor auch korrekt arbeitet, müssen Sie natürlich TRACE abschalten („[“-Taste).
Wenn Sie die Geschwindigkeit von TRACE verändern möchten oder das Programm in einen anderen Speicherbereich verlegen wollen, müssen Sie einen neuen Abgleich vornehmen. Es sei gleich gesagt, daß dies viel Geschicklichkeit erfordert.
Schreiben Sie hierzu ein kleines Programm, welches Ihnen eine Zeile vollschreibt und anschließend wieder löscht.
Nun müssen Sie versuchen, zuerst mit

```
1805:  20 B0 18   JSR  $18B0   |
1808:  8E 29 80   STX  $8029   |
180B:  8D 2A 80   STA  $802A   |
180E:  AD FA 03   LDA  $03FA   +
1811:  20 B0 18   JSR  $18B0   |
1814:  8E 2F 80   STX  $802F   | Accuinhalt
1817:  8D 30 80   STA  $8030   |
181A:  AD F9 03   LDA  $03F9   +
181D:  20 B0 18   JSR  $18B0   |
1820:  8E 32 80   STX  $8032   | X-Register
1823:  8D 33 80   STA  $8033   |
1826:  AD F8 03   LDA  $03F8   +
1829:  20 B0 18   JSR  $18B0   |
182C:  8E 35 80   STX  $8035   | Y-Register
182F:  8D 36 80   STA  $8036   |
1832:  AD FE 03   LDA  $03FE   +
1835:  20 B0 18   JSR  $18B0   |
1838:  8E 38 80   STX  $8038   |  Stapelzeiger
183B:  8D 39 80   STA  $8039   | (Stackpointer)
183E:  AD FB 03   LDA  $03FB   ┘
183E:  AD FB 03   LDA  $03FB   ┐
1841:  A2 10      LDX  =$10    |
1843:  2A         ROL          | Untersucht das
1844:  A8         TAY          | Statusregister
1845:  90 04      BCC  $184B   |
1847:  A9 1E      LDA  =$1E    | und setzt die
1849:  B0 03      BCS  $184E   | Anzeige
184B:  EA         NOP          |
184C:  A9 2D      LDA  =$2D    | (- gelöscht)
184E:  9D 3B 80   STA  $803B,X |
1851:  98         TYA          | (↑ gesetzt)
1852:  CA         DEX          |
1853:  CA         DEX          |
1854:  D0 ED      BNE  $1843   ┘
1856:  A2 0F      LDX  =$0F    ┐
1858:  A0 FF      LDY  =$FF    |Verzögerung 'grob'
185A:  AD 12 E8   LDA  $E812   |
185D:  C9 EF      CMP  =$EF    ← Prüft auf 'BREAK'
185F:  F0 33      BEQ  $1894   | (zurück in Monitor)
1861:  C9 FE      CMP  =$FE    ← Prüft auf 'REVERSE'
1863:  F0 04      BEQ  $1869   | (schaltet TRACE ein)
1865:  C9 FD      CMP  =$FD    ← Prüft auf '['
1867:  D0 03      BNE  $186C   | (schaltet TRACE ab)
1869:  8D FF 03   STA  $03FF   |
186C:  AD FF 03   LDA  $03FF   |
186F:  C9 FD      CMP  =$FD    |
1871:  F0 3A      BEQ  $18AD   ← Trace abgeschaltet,
1873:  88         DEY          | dann IRQ normal beenden
1874:  D0 E4      BNE  $185A   |
1876:  CA         DEX          |
1877:  D0 DF      BNE  $1858   ┘
1879:  A0 25      LDY  =$25    ┐
187B:  AD 12 E8   LDA  $E812   |
187E:  C9 7F      CMP  =$7F    ← Pruft auf '='
1880:  D0 03      BNE  $1885   | (ergibt schnelles TRACE
1882:  A0 24      LDY  =$24    |
1884:  EA         NOP          +
1885:  EA         NOP          |
1886:  EA         NOP          | Vergöserung 'fein'
1887:  EA         NOP          |
1888:  EA         NOP          | und 'sehr fein'
1889:  EA         NOP          |
188A:  EA         NOP          | (siehe Beschreibung)
188B:  EA         NOP          |
188C:  EA         NOP          |
188D:  EA         NOP          |
188E:  88         DEY          |
188F:  D0 FD      BNE  $188E   ┘
1891:  4C 7E E6   JMP  $E67E   ← IRQ ohne Tastaturabfrag
                                 beenden

1894:  68         PLA          ┐
1895:  68         PLA          |
1896:  68         PLA          |
1897:  68         PLA          | BREAK ausführen
1898:  68         PLA          |
```

```
1899:   68          PLA          |
189A:   A9 57       LDA  =$57    | Adresse
189C:   8D FC 03    STA  $03FC   | = low Byte
189F:   A9 19       LDA  =$19    |
18A1:   8D FD 03    STA  $03FD   | = high Byte
18A4:   A2 06       LDX  =$06    |
18A6:   BD F7 03    LDA  $03F7,X |  b. in Monitor
18A9:   48          PHA          |
18AA:   CA          DEX          |
18AB:   D0 F9       BNE  $18A6   ┘
18AD:   4C 85 E6    JMP  $E685   ← IRQ 'korrekt' beenden
18B0:   A8          TAY          ┐
18B1:   0A          ASL          | Subroutine:
18B2:   0A          ASL          |
18B3:   0A          ASL          |
18B4:   0A          ASL          | Umformung der Daten
18B5:   4A          LSR          | zur Anzeige
18B6:   4A          LSR          |
18B7:   4A          LSR          | das X-Register enthält
18B8:   4A          LSR          | zuletzt die 1. Stelle
18B9:   AA          TAX          |
18BA:   BD CA 18    LDA  $18CA,X | im Accu befindet sich
18BD:   48          PHA          | die 2. Stelle
18BE:   98          TYA          |
18BF:   4A          LSR          |
18C0:   4A          LSR          |
18C1:   4A          LSR          |
18C2:   4A          LSR          | zb. 0100 1011 wird als
18C3:   AA          TAX          |
18C4:   BD CA 18    LDA  $18CA,X | $4B angezeigt
18C7:   AA          TAX          |
18C8    68          PLA          |
18C9    60          RTS          ┘
18CA 30 31 32 33 34 35 36 37     *
18D2 38 39 01 02 03 04 05 06     *
18DA 20 20 10 03 20 20 20 01     *
18E2 03 20 18 12 20 19 12 20     * Daten
18EA 13 10 20 20 20 03 20 1A     *
18F2 20 CD 20 04 20 02 20 20     *
18FA 20 16 20 0E 20 20           *
```

der groben Verzögerung (Zeile 1856), den Vorgang zu verlangsamen. Mit der Zeile 1879 und dem Sprung in 1880 erfolgt dann der „Feinabgleich", bis die Anzeige steht. Der Abgleich für „schnelles Trace" („="-Taste) erfolgt in Zeile 1882.
Natürlich können Sie mit TRACE auch Basic-Programme verfolgen, und so die ROM-Routinen durchstöbern. Vergessen Sie dann aber nicht das Programm zuvor zu schützen.

Zum Schluß ein kleines Beispielprogramm.

```
  PC   AC XR YR SP   C Z I D B   V N
 0343  2A 08 00 8C   - 1 - - - 1 - -


.M 033A,0342
              0  1  2  3  4  5  6  7
.:  033A A9 20 49 0A A0 10 99 00
.:  0342 83 88 D0 FA F0 F4 EA EA
```

Wenn Sie das Programm mit G 033A (SYS 826) aufrufen, werden Sie zuerst nichts erkennen. Bei eingeschaltetem TRACE sehen Sie hingegen sofort, daß es sich um ein Lauflicht handelt. Mit BREAK können Sie dann das Programm abbrechen, sofern Sie die richtige Rücksprungadresse eingegeben haben (siehe Text).

Aufgrund des geänderten Betriebssystems sind beim CBM folgende Daten zu ändern:
Der Jump-Befehl in 17B2 weist auf den eingebauten Monitor. Diese Adresse kann auch beim STOP-Befehl (Zeile 189A,189F) verwendet werden.

```
17A0:   78          SEI
17A1:   A5 90       LDA  $90
17A3:   49 9B       EOR  =$9B
17A5:   85 90       STA  $90
17A7:   A5 91       LDA  $91
17A9:   49 F1       EOR  =$F1
17AB:   85 91       STA  $91
17AD:   58          CLI
17AE:   EA          NOP
17AF:   EA          NOP
17B0:   EA          NOP
17B1:   EA          NOP
17B2:   4C 11 FD    JMP  $FD11


1891:   4C E4 E6    JMP  $E6E4

18AD:   4C 2E E6    JMP  $E62E
```

Andreas Garzotto

Algebraisches-Operations-System (AOS) für HP-41 C

Beschreibung

Für den Taschenrechner HP-41C ist ein Programm zu schreiben, das eine Eingabeweise der Zahlen und Operationszeichen in der Art erlaubt, wie sie üblicherweise vorkommen, d.h. eine Rechnung wird in der Reihenfolge der Operationen von links nach rechts eingetastet und der Rechner berücksichtigt automatisch die Prioritäten der Rechnerstufen in der Ausführung. Ebenso ist es möglich, Klammern einzugeben und durch das Gleichheitszeichen alle noch nicht ausgeführten Operationen zu beenden. Insbesondere soll dieses Programm die Arbeitsweise der Taschenrechner TI-58 und TI-59 kopieren (9 Klammerebenen), die mit einem solchen System (Hardware), „AOS" genannt, ausgerüstet sind.

Da es möglich ist, dem HP-41C beliebige Funktionen einer Taste zuzuordnen, können die Rechenoperationen direkt durch die dazu vorgesehenen Tasten ausgeführt werden (+, −, / usw.). Das „=" wird durch die „Enter"-Taste ausgeführt, „Klammer auf" und „Klammer zu" wird den Tasten „A" und „B" zugeordnet, und gelöscht wird mit „C".

Um Zwischenergebnisse, die durch die Klammerebenen entstehen, abzuspeichern, sind acht Speicherregister erforderlich (das neunte Zwischenergebnis wird im X-Register aufbewahrt). Die Operationszeichen werden als Zahlen kodiert, und zwar entsprechend ihrer Prioritäten bei der Ausführung der Operationen. Für sie werden 15 Register benötigt (nicht nur acht, da ebenfalls die „Klammer auf"-Kodes hier abgespeichert werden).

Gibt man nun Zahlen und Zeichen ein, werden diese solange abgespeichert, bis ein Operationszeichen mit „niedrigerem Wert" als das vorangegangene eingegeben wird. Dann werden die abgespeicherten Zahlen vom zuletzt eingegebenen her solange mit den ebenfalls gespeicherten Operationen verknüpft, bis entweder eine Operation mit niedrigerer Priorität als der der zuletzt eingegebenen folgt oder ein „Klammer auf"-Zeichen gefunden wird oder alle gespeicherten Operationen ausgeführt sind. Wird die „Klammer zu"-Taste gedrückt, so werden alle Operationen ausgeführt bis zum nächsten „Klammer auf". Durch „=" wird alles Gespeicherte ausgeführt.

Beispiel

Aufgabe: $3+4\cdot(5-3)\cdot 2\uparrow(3\cdot 2)-500=?$ (15)

Vorbereitung: XEQ„AOS"

Tastenfolge: 3 + 4 × (|„A"| 5 − 3) |„B"| × 2 Y↑X (|„A"| 3 × 2) |„B"| − 500 = |„Enter"|

Statusinformationen

Speicherregister:

R00	Indirekter Zwischenergebniszähler
R01 – R08	Zwischenergebnisse
R09	Belegt
R10	Indirekter Zeichenzähler
R11 – R25	(Operations-)Zeichen
R26	Belegt

Das Programm arbeitet mit 9 Klammerebenen. Um die Anzahl zu vergrößern, müssen die oben beschriebenen Register 9, 10 und 26 entsprechend verlegt werden.

Flags:

F12	Gesetzt, wenn „=" ausgeführt wird
F13	Gesetzt, wenn „)" ausgeführt wird
F27	Immer gesetzt (User-Modus)

Labels: 00 – 11 A B C „+" „−" „·" „/" „↑" „=" „AOS"

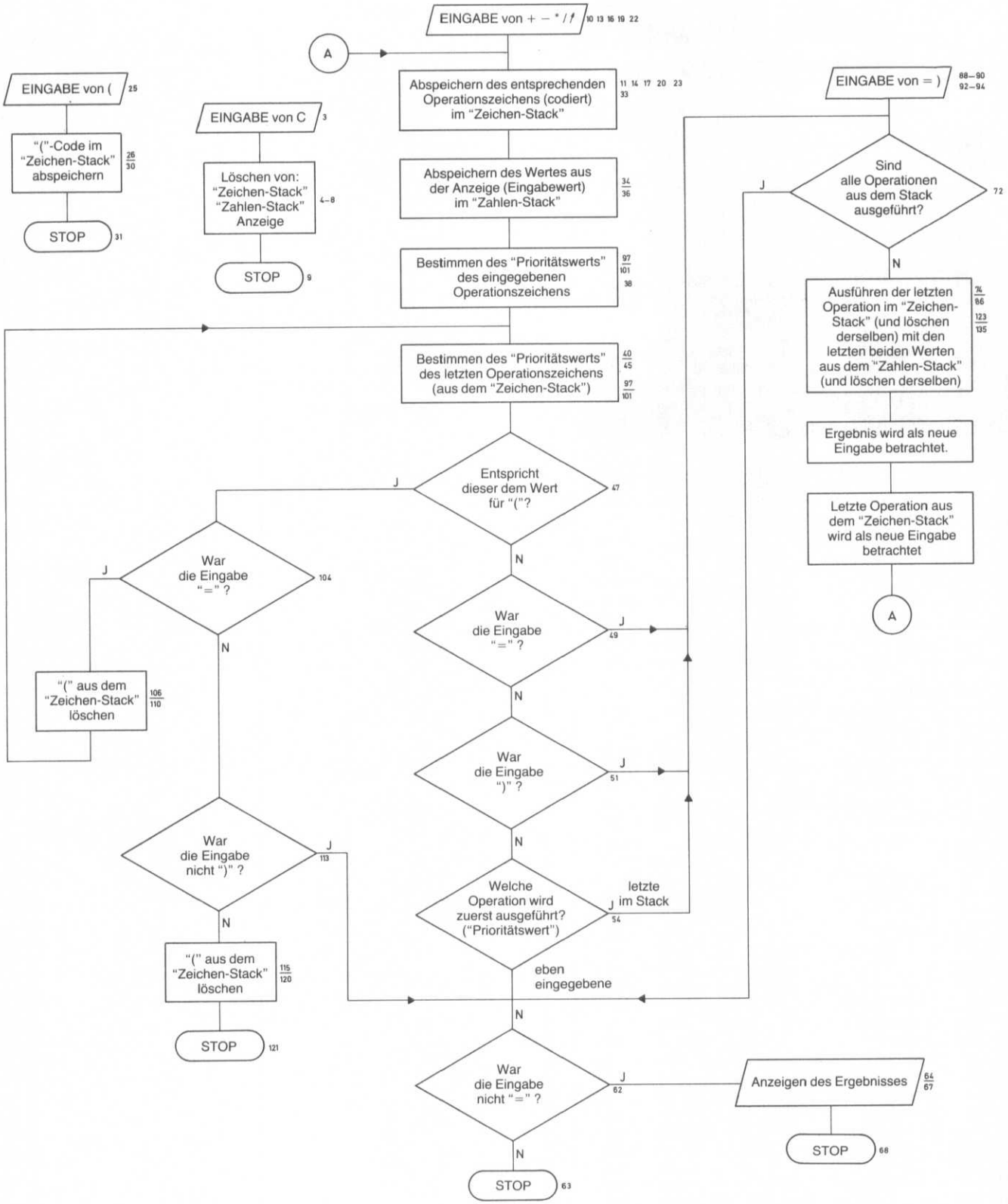
Tastenfeldzuordnungen:

Lbl„="	→	Enter↑
Lbl„+"	→	+
Lbl„−"	→	−
Lbl„×"	→	·
Lbl„/"	→	/
Lbl„↑"	→	y↑x

Automatische Zuordnungen:

Klammer auf	→	A
Klammer zu	→	B
Löschen	→	C

Speicherbereichsverteilung: mindestens SIZE 027



Nach jeder STOP-Anweisung ist das Programm bereit, nach einer Zahleneingabe bei einer EINGABE weiterzuführen.

Die Zahlen neben den Kästchen weisen auf die Zeilennummern hin, bei welchen die Funktion im Kästchen ausgeführt wird. Einige Zeilennummern sind nicht aufgeführt; auf den entsprechenden Zeilen sind Labels, Sprunganweisungen oder Vorbereitungsschritte zu vergleichen

(bitte umblättern)

Tölzer Straße 5
D–8150 Holzkirchen
Tel.: (0 80 24) 18 14

Gerätebau, Computersysteme-Software

Lieferung per NN oder Vorkasse auf Postscheckkto. 284 558–807 Mchn oder Euroscheck
Mindestbestellmenge bei Vorkasse DM 30,–, sonst DM 5,– Zuschlag.

MÜNZENLOHER GMBH

Color und Graphik
Das ATARI-System bietet schier unendliche Möglichkeiten für Farbdarstellung und programmierbare Graphik. Es gibt z.B. 128 unterschiedliche Farbmöglichkeiten (in der Kombination aus 16 Farben in jeweils 8 Abstufungen).

ATARI 400 PAL, 16K-RAM	DM	1.719,–
ATARI 800 PAL, 16K-RAM	DM	3.018,–
ATARI 800 PAL, 32K-RAM	DM	3.299,–
ATARI 800 PAL, 48 K-RAM	DM	3.634,–

Alle Preise incl. Netzteil, ATARI BASIC Buch (deutsch), Betriebsanleitung (deutsch), BASIC Computersprache (ROM-Modul), BASIC-Reference Manual (englisch).

System Zubehör:

ATARI 410 P – Programm Recorder –	DM	239,–
ATARI 810 – Disketten Stationen – (Mini-Floppy 5 1/4 ")	DM	1.769,–
ATARI 822, – Thermo Drucker – (40 Zchn.)	DM	1.335,–
ATARI 850 – Interface Modul –.	DM	667,–
1 Paar Steuerknüppel CX 40-04	DM	79,–
1 Paar Drehregler CX 30-04	DM	79,–

Wir bieten außerdem folgende Programme für die ATARI-Computer an:

Best.-Nr.	Bezeichnung		Preis	
CX4104	Mailing List	Cass.	DM	69,–
CX4109	Graph IT	2 Cass.	DM	67,–
CX4103	Statistics I	Cass.	DM	67,–
CX4101	An Invitation to Progr. I	Cass.	DM	67,–
CX4107	Biorythm	Cass.	DM	49,–
CXL4007	Music Composer	ROM-Mod.	DM	199,–
CX4110	Touch Typing	2 Cass.	DM	84,–
CX4121	Energy Czar	Cass.	DM	49,–
CX4108	Hangman	Cass.	DM	49,–
CX4102	Kingdom	Cass.	DM	49,–
CX4112	States & Capitals	Cass.	DM	49,–
CX4114	Europ.Countr. & Capitals	Cass.	DM	49,–
CXL4004	Basketball	ROM-Mod.	DM	134,–
CX4105	Blackjack	Cass.	DM	49,–
CXL4009	Computer Chess	ROM-Mod.	DM	134,–
CX4111	Space Invaders	Cass.	DM	67,–
CXL4011	Star Raiders	ROM-Mod.	DM	199,–
CXL4006	Super Breakout	ROM-Mod.	DM	134,–
CXL4010	3-D Tic-Tac-Toe	ROM-Mod.	DM	134,–
CXL4005	Video Easel	ROM-Mod.	DM	134,–
Programmsprachen				
CXL4003	Assembler/Editor	ROM-Mod.	DM	199,–
CXL4002	ATARI-BASIC	ROM-Mod.	DM	199,–

Fordern Sie unseren Gesamtkatalog sowie die neuesten ATARI-Sonderprospekte an!

Programmierhilfen

Algebraisches Operations-System (AOS) für HP-41C

```
01♦LBL "AOS"
02 SF 27
03♦LBL C
04 1
05 STO 00
06 11
07 STO 10
08 CLX
09 STOP
10♦LBL "+"
11 1
12 GTO 00
13♦LBL "-"
14 2
15 GTO 00
16♦LBL "*"
17 3
18 GTO 00
19♦LBL "/"
20 4
21 GTO 00
22♦LBL "↑"
23 5
24 GTO 00
25♦LBL A
26 0
27 STO IND 10
28 1
29 ST+ 10
30 RCL Z
31 STOP
32♦LBL 00
33 STO IND 10
34 RDN
35 STO IND 00
36 P↑
37 XEQ 06
38 STO 09
39♦LBL 11
40 1
41 ST- 10
42 RCL IND 10
43 1
44 ST+ 10
45 RDN
46 XEQ 06
47 X=0?
48 GTO 08
49 FS? 12
50 GTO 07
51 FS? 13
52 GTO 07
53 RCL 09
54 X<=Y?
55 GTO 07
56♦LBL 09
57 RCL IND 00
58 1
59 ST+ 00
60 ST+ 10
61 RDN
62 FC?C 12
63 STOP
64 1
65 ST- 00
66 ST- 10
67 RDN
68 STOP
69♦LBL 07
70 RCL 00
71 1
72 X=Y?
73 GTO 09
74 RCL IND 10
75 STO 26
76 1
77 ST- 10
78 RCL IND 10
79 RCL IND 00
80 1
81 ST- 00
82 RDN
83 RCL IND 00
84 X<>Y
85 XEQ IND Z
86 RCL 26
87 GTO 00
88♦LBL "="
89 SF 12
90 STO IND 00
91 GTO 07
92♦LBL B
93 SF 13
94 STO IND 00
95 GTO 07
96♦LBL 06
97 ENTER↑
98 ENTER↑
99 2
100 MOD
101 +
102 RTN
103♦LBL 08
104 FC? 12
105 GTO 10
106 RCL IND 10
107 1
108 ST- 10
109 RDN
110 STO IND 10
111 GTO 11
112♦LBL 10
113 FC?C 13
114 GTO 09
115 RCL IND 10
116 1
117 ST- 10
118 RDN
119 STO IND 10
120 RCL IND 00
121 STOP
122♦LBL 01
123 +
124 RTN
125♦LBL 02
126 -
127 RTN
128♦LBL 03
129 *
130 RTN
131♦LBL 04
132 /
133 RTN
134♦LBL 05
135 Y↑X
136 RTN
137 .END.
```

Dietmar Wäber

Ausgabe der Symboltabelle für AIM-65/PC-100

Beschreibung

Der erste Durchlauf des Assemblers erzeugt die sogenannte Symboltabelle. In dieser Tabelle werden die Namen der verschiedenen Symbole und deren Werte gespeichert. Im zweiten Pass ersetzt der Assembler die symbolischen Namen durch die in der Tabelle stehenden Werte und erzeugt somit den Objektkode in hexadezimaler Form.
Da die Symboltabelle wichtige Informationen enthält, wird sie druckaufbereitet am Ende eines Programmlistings ausgegeben. Im 4K-Assembler-ROM von Rockwell hat diese Option leider keinen Platz mehr gefunden.
Mit den hier aufgezeigten Verfahren kann man sich jedoch die alphanumerisch sortierte Symboltabelle ausgeben lassen. Zusätzlich zu den hexadezimalen Werten wird auch noch der betreffende Dezimalwert ausgedruckt.
Nach Initialisierung des Assemblers mit der Taste <N> wird man zur Eingabe der Anfangsadresse der Symboltabelle aufgefordert. Diese Adresse wird in den Speicherzellen $3A_H$ und $3B_H$ abgelegt. Nach Ablauf der Assemblierung steht in Speicherzelle $\emptyset C_H$ die Anzahl der verschiedenen Symbole. Das Programm beschränkt sich auf die Verarbeitung von 256 unterschiedlichen Namen.
Zum Aufbau der Tabelle siehe Bild 1.

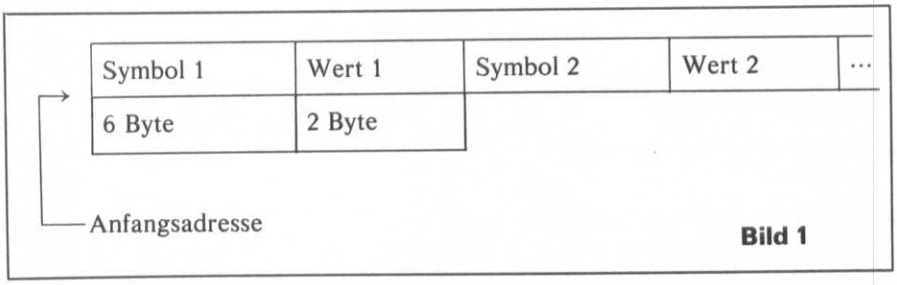


Bild 1

```
 10 DIM H$(15),S$(PE
EK(255)+1)
 20 DATA 0,1,2,3,4,5
,6,7,8,9,A,B,C,D,E,F
 30 FOR I=0 TO 15
 40 READ H$(I)
 50 NEXT I
 60 A$="------------
--------"
 70 PRINT A$
 80 PRINT"XREF - SYM
BOLTABELLE
 90 PRINT A$
 100 PRINT" " : PRIN
T" "
 110 PRINT"SYMBOL  H
EX  DEZ
 115 PRINT " "
 120 Z=PEEK(255)
 130 A=PEEK(254)+256
*PEEK(253)
 140 FOR I=A TO A+8*
Z
 150 SY$=SY$+CHR$(PE
EK(I))
 160 K=K+1
 170 IF K<8 THEN 220
 180 K=0
 190 L=L+1
 200 S$(L)= SY$
 210 SY$=""
 220 NEXT I
 300 FOR I=1 TO Z
 310 K=I
 320 FOR J=I+1 TO Z
 325 IF J>Z THEN 340
 330 IF S$(J)<S$(K)
THEN K=J
 340 NEXT J
 350 IF K=J THEN 390
 360 SY$=S$(I)
 370 S$(I)=S$(K)
 380 S$(K)=SY$
 390 NEXT I
 400 FOR I=1 TO Z
 410 SY$=S$(I)
 420 D1=ASC(MID$(SY$
7,1))
 430 D2=ASC(MID$(SY$
8,1))
 440 D=D1*256+D2
 450 H1=INT(D1/16)
 460 H2=D1-H1*16
 470 H3=INT(D2/16)
 480 H4=D2-H3*16
 490 HEX$=H$(H1)+H$(
H2)+H$(H3)+H$(H4)
 500 PRINT MID$(SY$,
1,6);TAB(8);HEX$;TAB
(12);D
 510 NEXT I
 520 PRINT"******
 530 PRINT" " : PRIN
T" "
 540 PRINT A$
 550 PRINT"SYZAHL=";
Z
 560 PRINT"SYANF. =";
A
 570 PRINT A$
 600 PRINT" " : PRIN
T " "
 999 END
```

Bild 2

Jedes Tabellenelement benötigt also 8 Byte Speicherplatz.
Das Hauptprogramm (Bild 2) übernimmt die vom Assembler gelieferten Informationen und verarbeitet sie.
Da der BASIC-Interpreter die Werte des Assemblers in der Zeropage überschreiben würde, muß man vor Eintritt in das BASIC-System das kleine Assemblerprogramm in Bild 3 laufen lassen. Dieses Programm bringt den Inhalt der wichtigen Speicherzellen in einen Bereich auf der Zeropage, den BASIC nicht benutzt. Das Programm kann durch Betätigen der Taste <F1> gestartet werden.
BASIC überschreibt ferner den gesamten freien RAM-Bereich ab Adresse $\emptyset 214_H$ mit AA_H. Um die Zerstörung der noch benötigten Tabelle zu verhindern, muß man die Startadresse der Symboltabelle möglichst in den obersten Bereich des freien Speichers legen. Dieser Bereich kann durch entsprechende Wahl der Memory Size geschützt werden.
Ein auf diese Weise erzeugtes Listing der Symboltabelle zeigt Bild 4. Die Symbole stammen aus dem Hilfsprogramm zum Sichern der Speicherstellen.

(bitte umblättern)

CHIP-BÖRSE

Unter dieser Rubrik veröffentlicht CHIP Textanzeigen zu einem besonders günstigen Preis.

Nutzen Sie den Anzeigen-Service, wenn Sie

- Hardware oder Software verkaufen wollen
- Programme austauschen möchten
- Kontakte und Erfahrungsaustausch anstreben
- Literatur suchen oder anbieten

Private Gelegenheitsanzeigen je Druckzeile 6 DM inkl. MwSt.

Musteranzeige

Suche Mini-Printer, Ansteuerelektronik, möglichst 64 Zeichen/Bit parallel Eingang/Zeichen seriell. H. J. Kraft, S 17/68 Mannheim

nur 24 DM

Gewerbliche Gelegenheitsanzeigen je Druckzeile 8 DM zuzügl. MwSt.

Musteranzeige

Verkaufe Datensichtgeräte 80 × 24 Z, VB 750,– DM. Mikrocomputer-Kits, Peripherie-Software äußerst günstig. H. Jung, Telefon (0 40) 31 46 78

nur 32 DM zuzügl. MwSt.

Chiffregebühr 6 DM inkl. MwSt.

CHIP

Anzeigenservice
Vogel-Verlag
Postfach 67 40

D-8700 Würzburg 1

Ausgabe der Symboltabelle für AIM-65/PC-100-Assembler

```
<N>
ASSEMBLER
FROM=0F00  TO=1000
IN=M
LIST?Y
LIST-OUT=P

OBJ?N
PASS 1

PASS 2


==0000 ENTRY=*

==0000 BASIC=$B000

==0000 SYZAHL=$0C

==0000 STABH=$3A

==0000 STABL=$3B

==0000 ZAHL=255

==0000 SAVEH=254

==0000 SAVEL=253

==0000
       *=$010C
==010C
4C0800 JMP MOVE
==010F
       *=0200
==00C8 MOVE=*
A50C   LDA SYZAHL
85FF   STA ZAHL
A53A   LDA STABH
85FE   STA SAVEH
A53B   LDA STABL
85FD   STA SAVEL
4C00B0 JMP BASIC
       .END INIT
 ERRORS= 0000
```

Bild 3

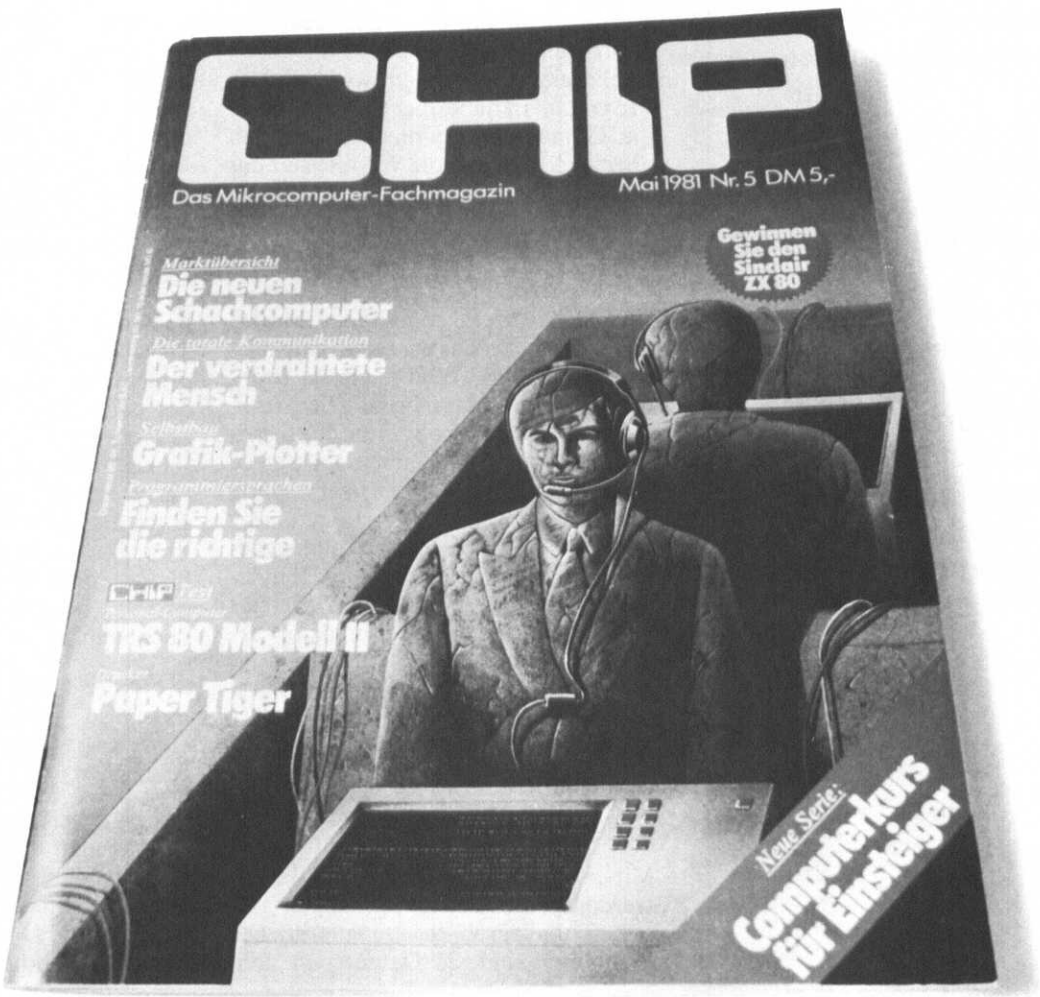
```
<C>
MEMORY SIZE? 3500
WIDTH? 15
 2970 BYTES FREE
 SIEMENS PC100 BASIC

                  RUN
--------------------
XREF - SYMBOLTABELLE
--------------------


SYMBOL   HEX   DEZ

BASIC    B000 45056
ENTRY    0000 0
MOVE     00C8 200
SAVEH    00FE 254
SAVEL    00FD 253
STABH    003A 58
STABL    003B 59
SYZAHL   000C 12
ZAHL     00FF 255
******


--------------------
SYZAHL= 9
SYANF. = 3840
--------------------
```

Bild 4

Hier nun einige Anmerkungen zum Programm:

Zeilennummer	Bemerkung
30— 50	Aufbau einer Sedezimaltabelle
60—115	Drucken der Überschriften
120	Übergabe der Symbolzahl
130	Berechnung der Anfangsadresse der Tabelle
140	Schleife über die Speicherzellen der Tabelle
150—170	Aufbau eines Symbolelementes in SY $
180—210	Speicherung dieses Elementes im Vektor S $(*)
300—390	alphanumerische Sortierung des Vektors
420	D1 := higher Byte des Symbolwertes
430	D2 := lower Byte des Symbolwertes
440	Berechnung des Dezimalwertes aus D1 u. D2
450—490	Wandlung des Wertes in hexadezimale Form
500	Ausgabe des Symboles
550	Ausgabe der Anzahl von Symbolen
560	Ausgabe der Startadresse der Symboltabelle

CHIP, Zeitschrift für Mikrocomputer-Technik. Spiegelbild der ganzen Mikrocomputer-Welt. CHIP bringt Ausgabe für Ausgabe alles, worauf es ankommt. Für alle, die mehr wissen wollen und mehr wissen müssen, wenn es um Mikrocomputer-Technik geht.



CHIP gibt es
beim Zeitschriftenhandel.
Oder direkt bestellen
beim CHIP-Leserservice.

Vogel-Verlag, D-8700 Würzburg 1, Postfach 67 40.

Computerspiele garantieren den Spaß mit dem Computer. Es gibt Nonsens-Spiele und solche, die Geist und Intelligenz fordern. Die Auswahl in diesem Heft beschränkt sich bewußt auf anspruchsvollere Spiele, die auch noch manchen Lerneffekt einschließen. Auf den nächsten Seiten finden Sie folgende Spiele:
— Der verflixte Würfel, in PASCAL programmiert,
— Mühleprogramm für alle BASIC-Rechner ab 32K,
— Würfelspiel mit dem TI-59/PC 100C,
— ELIZA II mit ausführlicher Beschreibung für 6502-Systeme.
Außer dem Würfelspiel sind alle Spiele auf eine Vielzahl von Rechnern übertragbar und damit „spielerische" Lehrbeiträge zur Programmierung.

Johannes Leckebusch

Der verflixte Würfel

Beschreibung

Grundlage für ein computergerechtes Modell des Rubic-Cube, geschrieben in PASCAL, ist die intensive Nutzung von Backtracking. Für ein gegebenes Würfelproblem soll die kürzestmögliche Rückdrehung gefunden werden. Das Programm ist dabei nicht so schnell wie die Guiness-Champions, es läßt sich aber auch nicht wie die Weltmeister der skurilen Rekorde von Intuition leiten, sondern nutzt mathematische Algorithmen.

Das Programm ist in zwei Teile gegliedert: „Simulator" enthält das Modell des Würfels, auf das wir durch eine Prozedur „ro" zugreifen können, um Drehoperationen auszuführen. „Backtrakking plus" ist ein Programm, das dieses Modell benutzt, um einen verbesserten Backtracking-Algorithmus zur Lösung eines Würfelproblems auszuführen. (Vgl. die im Programmkopf angegebenen Artikel über Backtracking.) Ein Zugriff auf das Würfelmodell ist dabei nur über die Prozeduren in „Simulator" erlaubt. Zunächst werden Initialisierungs-Prozeduren aufgerufen. Durch „display (false)" kann ein bestimmter Zustand des Würfels als Problem eingegeben werden, „display (true)" schreibt dagegen den Zustand des Modells auf den Bildschirm.

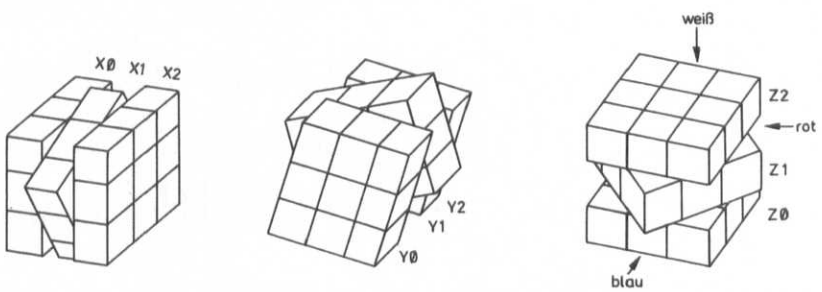
Mit „ro (instruktion)" und „test (level)" werden verschiedene Drehungen ausgeführt und deren Resultat geprüft. Die Parameter sind als Datentypen definiert, „tinstruktion" enthält die Drehachse („achse"), den Index einer der drei dadurch bestimmten Ebenen („achsindex") und das Vorzeichen der Drehung („vorzeichen") als „plus", „hundert" oder „minus". Die Komponente „umkehr" wird normalerweise auf „false" gesetzt. Läßt man sie aber „true" sein, so kann dieselbe Drehung umgekehrt werden. Hiervon macht die Prozedur „backtrack" Gebrauch, um das Ergebnis erfolgloser Versuche wieder rückgängig zu machen.

In Bild 1 sehen Sie, wie die einzelnen Teilwürfel durch ein kartesisches Koordinatensystem erfaßt werden. Je drei der neun möglichen Drehebenen bezeichnen wir nach der Achse, auf der sie senkrecht stehen, als x-, y- oder z-orientierte Ebene und unterscheiden sie durch den Index 0···2. Eine elementare Rotation, das ist die Drehung an einer Ebene um 90, 180 oder 270° (im Uhrzeigersinn), bezeichnen wir für die 0te x-Ebene mit

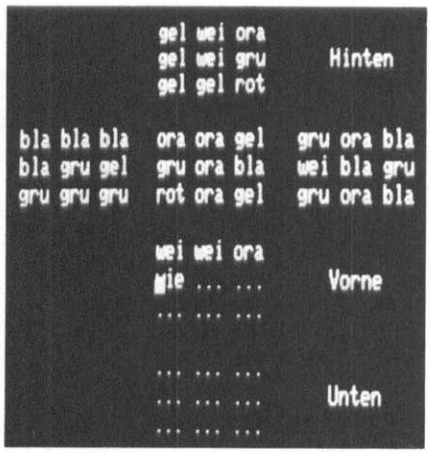
rx0p, rx0h oder rx0m.

Es ist also — rx0p = rx0m. Wir sind so in der Lage, jede Drehung rückgängig zu machen.

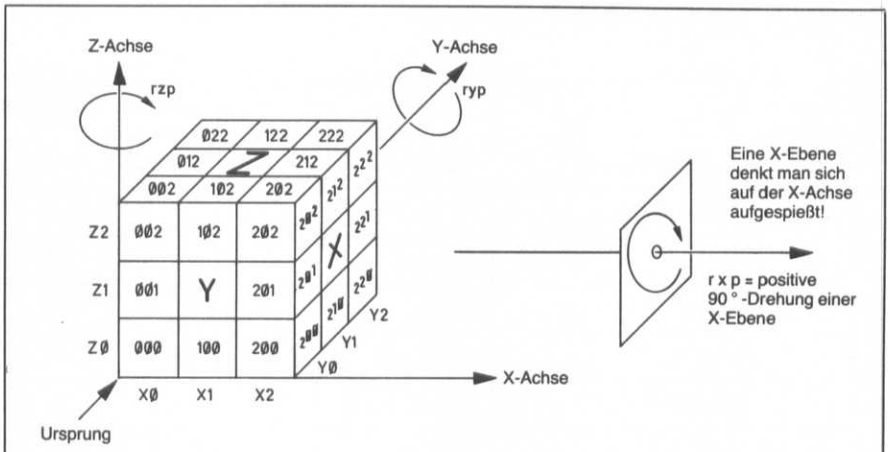
Wir beschreiben den Würfel als dreidimensionalen Array aus Teilwürfeln, von denen jeder drei Seiten besitzt (x, y, z) — von denen in der gewohnten Darstellung (Bild 1) mindestens eine, höchstens drei sichtbar sind.



"Hier sieht man, was mit den drei X-Ebenen (links), Y-Ebenen (mitte) und den drei z-Ebenen gemeint ist.
Der Würfel befindet sich jeweils in Normal-Stellung, weiß oben - blau vorne - rot rechts."



Display während der Eingabe eines Würfelproblems. Bei der letzten Eingabe wurde "wei" als "wie" vertippt, daher springt der Cursor zum Anfang der Eingabeposition zurück.



So ist der Würfel ins Koordinatensystem "eingeheftet". Dieses bleibt immer starr, es dreht sich nicht mit, auch wenn der Würfel als Ganzes gekippt wird!

Vergleichen Sie dies mit den Typ-Definitionen im Programmkopf (tfarbe bis tkubus).

Bei einer Drehung geschehen zwei Dinge: Die Teilwürfel einer Ebene wechseln ihre Plätze und ihre Seitenflächen die Orientierung. Bei der Drehung einer x-Ebene bleiben die Flächen mit x-Orientierung x-Flächen, die beiden anderen tauschen ihre Lage untereinander. Wenn ein Teilwürfel weniger als drei sichtbare Flächen besitzt, nehmen wir an, daß er eine entsprechende Anzahl von „Füllflächen" hat, denen wir die Farbe „Schwarz" zuordnen. Der x-Index eines Teilwürfels wird bei einer x-Drehung nicht verändert. Schreibt man die beiden anderen Indizes nach ihrer Reihenfolge in der x-y-z-Notation als yi, zi auf und die Werte, die sie nach einer Operation „rx0p" erhalten, als yj, zj, so beschreibt folgende Formel die Wanderung der Teilwürfel:

yj = zi; zj = 2 − yi;

Wir können jede Elementaroperation durch ein-, zwei- oder dreimaliges Ausführen dieser einen Drehung erzeugen.

Betriebsanleitung für das Programm

Gehen Sie bei der Eingabe besonnen vor. Kommentare in {}-Klammern dürfen Sie weglassen, fügen Sie aber die Compiler-Anweisung {$I−} ein! Sie können "{" durch "(*" und "}" durch „*)" ersetzen (ziehen Sie ggf. das Pascal-Handbuch Ihres Rechners zu Rate). Falls Sie kein UCSD-Pascal besitzen, ersetzen Sie ggf. die Anweisung „exit (name)" durch ein „goto" zu einem Label am Ende der Prozedur, deren Name in „(name)" erscheint.
Wenn Sie das Programm wie abgedruckt eingegeben und compiliert haben, verfügt es noch nicht über die Fähigkeit, den Backtracking-Algorithmus abzukürzen. Beim Start schreibt es die Meldung „Diskfehler" auf den Bildschirm. Dies ist normal. Das Programm sollte nun eine Abwicklung des Würfels mit den der Endstellung entsprechenden Farbnamen (auf drei Zeichen gekürzt) auf den Bildschirm schreiben und ein „Menue" anbieten. Sie treffen Ihre Wahl, indem Sie die Taste mit dem Buchstaben drücken, der durch eine Klammer (hier wäre es „K") abgesetzt ist. Falls das Programm mit einer I/O-Fehlermeldung abbricht, haben Sie die Anweisung {$I−} vergessen oder müssen diese anders formulieren. Der Grund hierfür ist, daß das Programm einen Informationsfile auf der Systemdiskette sucht, der nicht existiert, weil es ihn erst selbst erzeugen soll. „{$I−}" weist das System an, I/O-Fehler zu ignorieren.

Bitte führen Sie als erstes folgenden Test durch, ohne ein Würfelproblem einzugeben: Drücken Sie „R(un". Das Programm muß nun nach sechs Versuchen folgende „Lösung" präsentieren:

rx0p, rx1p, rx2p;

Danach erscheint der Würfel auf dem Display gegenüber der Ausgangsstellung als Ganzes verdreht, ist aber korrekt.

Wenn diese Prüfung einwandfrei verläuft, können Sie ein Problem mit I(nput eingeben, indem Sie entsprechend der Stellung des Cursors die drei Anfangsbuchstaben der Farben der Teilwürfelfläche eintippen. Bei unkorrekten Eingaben verweilt der Cursor auf der laufenden Position. Falls sich die beim Programmstart gezeigte Abwicklung des Würfels nicht auf Ihren Würfel abbilden läßt, müssen Sie die Prozedur „farbinit" entsprechend ändern. Beispielsweise seien bei Ihrem Würfel (dunkel-)rot und orange vertauscht. Dann müssen Sie in Farbinit schreiben:

„farbtabelle [rot] := „ora";
farbtabelle [orange] := ‚rot';".

Wenn Sie merken, daß Sie falsche Farbflächen eingegeben haben, können Sie die Eingabe verlassen, indem Sie dreimal die Leertaste drücken.

Beginnen Sie mit einfachen Problemen, indem Sie den idealen Würfel nur zwei- oder dreimal verdrehen. In welcher Lage Sie den Würfel eingeben, ist gleichgültig, aber: Lassen Sie nach der Eingabe den Würfel so auf dem Tisch liegen, daß die Fläche in der Kreuzungsmitte des Displays oben und die mit „vorne" bezeichnete vorn steht. Genau von dieser Lage aus führen Sie dann die vom Programm berechneten Züge aus, wobei Sie nur die jeweils bezeichnete Ebene drehen.

Sie werden feststellen, daß das Programm sehr lange brauchen kann, um

(bitte umblättern)

Der verflixte Würfel

Statistik des Cube 1.1-Programms

Backtracking Zugtiefe n	Anzahl Versuche	Rechenzeit	„Backtracking plus" Versuche günstig	ungünstig	Zeit	Faktor	
1	27	0,7 bis 19 s	1	27	19 s	1	
2	675	8 min	11	45	31,5 s	15	1.7
3	16 227	3,16 h	24	93	1′ 5″	175	2.1
4	389 475	3,16 Tage	28	327	3,8 min	1191	3.5
5	9 347 427	10,8 Wochen	116	1746	20 min	5353	5.3
6	$2{,}2433 \cdot 10^8$	5 Jahre	?	?	8 h	(5353)	
7	$5{,}3840 \cdot 10^9$	120 Jahre	?	?	8 Tage	(5353)	
8	$1{,}2921 \cdot 10^{11}$	2880 Jhr.	?	?	7 Mt.	(5353)	
9	$3{,}1012 \cdot 10^{12}$	69 120 Jhr.	?	?	13 Jahre	(5353)	
10	$7{,}4429 \cdot 10^{13}$	1 658 880 Jhr.	?	?	309 Jhr.	(5353)	

berechnet!

die Lösung zu finden. Wenn Sie seine Leistungsfähigkeit verbessern wollen, so benutzen Sie die Betriebsart „K(ardlist" und lassen das Programm einige Stunden laufen. Es beginnt von der Idealstellung des Würfels aus alle möglichen Drehungen zu erproben und zählt die Anzahl verschiedener Farben auf den Seiten. Dieses Kriterium benutzt es später, um zu entscheiden, ob aus einer gegebenen Situation heraus der Versuch, die Rückdrehung in n Versuchen zu erproben, sinnvoll ist. Bitte fangen Sie damit an, das Programm die Zugtiefen von „Startlevel" = 1 bis „Maxlevel" = 2 oder 3 untersuchen zu lassen. Für Level 3 müßte das Programm bereits einige Stunden rechnen. Je nach Rechner und Programmsystem dauert das Ausprobieren und Bewerten einer Operation 0,7 bis 1,5 s. Es genügt aber, wenn Sie es laufen lassen, bis es bei der ersten Verdrehung die Parameter „x1m" abgeschlossen hat, denn vermutlich sind alle folgenden Zustände nur Symmetrien zu den bereits untersuchten. Das Programm speichert alle 500 Versuche als Resultat auf der Systemdiskette ab. Sie können es dann mit „Reset" oder „Power OFF—ON" abbrechen. Falls Sie einen Drucker besitzen, können Sie nach dem Neustart des Programms die erhaltene Liste mit „L(istkardfile" ausdrucken (ggf. erscheint sie auf dem Bildschirm).

Leider kann ich in dem verfügbaren Rahmen das Prinzip dieser Verbesserung (die bis zum Faktor 5300 geht) nicht näher diskutieren. Trotz allem schiebt dies das Ansteigen der Rechenzeit mit 27 hoch n (n = Anzahl zu prüfender Züge) nur hinaus. Offenbar braucht man mehr Kriterien oder ein gänzlich anderes Verfahren. Vielleicht machen Sie sich Gedanken darüber, wie man das vorgeführte Würfelmodell dazu benutzen kann, etwa das vom „Spiegel" veröffentlichte Lösungsverfahren zu programmieren. Dies liefert zwar nicht die kürzestmögliche Rückdrehung, aber es benötigt nicht soviel Zeit, um eine Lösung zu entdecken.

Es gibt Anzeichen dafür, daß eine Vorausberechnung der Verteilung nach fünf Zügen keine zusätzliche Verbesserung bringt!

Links neben der Zugtiefe „n" die Anzahl der Versuche ($27 + 24^2 + 24^3 + \cdots + 24^{n-1} + 24^n$) und die Rechenzeit mit Backtracking (im ungünstigsten Fall). Rechts die Anzahl Versuche, die das Programm für systematisch ausgewählte „günstige" und „ungünstige" Probleme brauchte, nachdem es die Verteilungen bis Level 3 einschl. des ersten Zuges „x1m" sowie etwa acht Stunden lang die Verteilungen nach vier Zügen untersucht hatte (etwa: 40 000 Versuche). Danach wächst die Anzahl Versuche wieder mit 24^n. Wie weit eine Einbeziehung der Verteilungen höherer Level den Zeitanstieg bremst, wurde noch nicht untersucht. Da der Gewinn durch n-Züge-Vorausschau linear abnimmt, bringt das Prinzip vermutlich nur bis vier Züge Vorausschau einen Gewinn.

```
{05. 03. 1981   Version 1.1m  System: CP/M - Pascal/M - TRS-80 Model II}
      {kompatibel mit UCSD-PASCAL}
{C U B E   B A C K T R A C K I N G  - Version 'CHIP 1.0'}

{ (C) by Johannes Leckebusch, Muenchen 1981}
{Begonnen:      20. 01. 81}
{Literatur:     Spiegel Nr. 4/35 v. 19. Januar 1981,
                'RUBIK'S CUBE' von Prof. Dr. Kurt Endl, Wuerfel Verlag GmbH
                Bild der Wissenschaft, November, Dezember 80
                CHIP SPEZIAL Nr. 2 bzw. CHIP Nr. 6 u. 7 1980:
                  "So funktioniert Backtracking"}

{$I-    I/O - Fehlerbehandlung im Programm}

program back;

const   clendline       = 00;           {Bitte entnehmen Sie den Hand-}
        clendscreen     = 26;           {buechern Ihres Systems die ent-}
        chome           = 30;           {sprechenden Steuercodes!!}
        czeilen         = 72;           {Papier - Format, DIN-A-3 Format}
        konsol          = 'console:';   {Hier Ausgabegeraet angeben}
        printer         = 'printer:';   {Hardcopy - Geraet, ev. console:}
        cmaxn           = 20;   {Maximale Zuganzahl, die untersucht wird}
        clistname       = 'kardvert.dat';      {Infofile fuer 'test'}

type    {primaere Datentypen fuer Wuerfel - Beschreibung}

        tfarbe          = (weiss, orange, gelb, rot, blau, gruen, schwarz);
        tfarbtabelle    = array [tfarbe] of string [3];
        tindex          = 0..2;
        twuerfel        = {packed} record x, y, z: tfarbe end;
        tebene          = array [tindex, tindex] of twuerfel;
        tkubus          = array [tindex, tindex, tindex] of twuerfel;

        {Datentypen fuer die Cubula - Definition - Subset}

        tsym            = (plus, hundert, minus,
                           xachse, yachse, zachse);
        tdimindex       = xachse..zachse;
        tvorzeichen     = plus..minus;
        tinstruktion    = (packed)        record  umkehr: boolean;
                                                  achse: tdimindex;
                                                  achsindex: tindex;
                                                  vorzeichen: tvorzeichen;
                                        end; {einfache Instruktion}
        tposition       = (packed)      record xi, yi, zi:    tindex end;
        lint            = integer;  {ggf. Long-integer / fuer Zugzaehler}
        tmaxl           = 0..6;         {Farben u. Bewertungslevels}
        tlevelset       = set of tmaxl;{Menge der Bewertungen}
        tkardvert       = array [1..6] of tmaxl;
        tfarbmeng       = set of tfarbe;
        tfarbt          = array [1..6] of tfarbmeng;
        tpointer        = ^tlistzeil;
        tlistzeil       = record liner:         tpointer;
                                 zeil:          tkardvert;
                                 levs:          tlevelset;
                          end;
{Anm.: Gepackte Records fuehren zu langsamerer Programmausfuehrung!}

var     kubus:                          tkubus;
        farbtabelle:                    tfarbtabelle;
```

Der verflixte Würfel

```
        sw:                              char;
        drucker:                         text;          (Hardcopy - File)
        listbas, lesepoin, neupoin:      tpointer;
        kardfile:                        file of tlistzeil;
        startlev, maxlev:                integer;
        maxlevel:                        integer;
        versuch:                         lint;
        protlist:                        array [1 .. cmaxn] of tinstruktion;
        ix, iy:                          tindex;
        farbt:                           tfarbt;
        t, zz:                           integer;
        ta:                              tkardvert;
        level:                           tlevelset;
        genkard, fertig:                 boolean;
        dummy:                           tinstruktion;

(25. 02. 1981   Version 1.1)

(C U B E   S I M U L A T O R  &  GLOBALE UTILITIES (Subset fuer Backtrack))

(System - I/O - Utilities ************************************************)

  procedure clearscreen;
  begin write(chr(chome), chr(clendscreen));
  end;

  procedure prompt (nachricht: string);
  begin write(chr(chome), chr (clendline), nachricht);
  end;

(SIMULATOR - Prozeduren **************************************************)

  procedure farbinit;                    (initialisiert Farbtabelle)
  (Diese Tabelle kann vom Benutzer auch anders initialisiert werden,
   wenn ein abweichend ausgefuehrter Rubik-Wuerfel verwendet wird)

  begin
    farbtabelle [weiss] :=      'wei'; farbtabelle [orange] :=     'ora';
    farbtabelle [gelb] :=       'gel'; farbtabelle [rot] :=        'rot';
    farbtabelle [blau] :=       'bla'; farbtabelle [gruen] :=      'gru';
    farbtabelle [schwarz] :=    '...';
  end (farbinit);

  procedure initschwarz;
  (Initialisiert die 'don't care' - Seiten)

  var   ix, iy, iz:     tindex;

  begin
    for ix := 0 to 2 do
      for iy := 0 to 2 do
        for iz := 0 to 2 do
          with kubus [ix, iy, iz] do
            begin x := schwarz; y := x; z := x end;
  end (initschwarz);

 procedure gutinit;
 (Stellt Idealzustand des Wuerfels her)

    procedure initbunt;
    (Initialisiert die Farbflaechen)
```

```
  var         ix, iy:         tindex;

  begin
    for ix := 0 to 2 do
      for iy := 0 to 2 do begin
        kubus [ix, iy, 2].z := weiss; kubus [ix, iy, 0].z := gelb;
        kubus [0, ix, iy].x := orange; kubus [2, ix, iy].x := rot;
        kubus [ix, 0, iy].y := blau; kubus [ix, 2, iy].y := gruen;
      end;
  end {initbunt};

begin {autinit}
  initschwarz; initbunt;
end {autinit};

procedure ro(param: tinstruktion);      {Dreht eine Schicht}
{Prinzip: Achse und Index werden uebergeben. Dadurch wird eine Ebene
 in Kubus definiert. Diese wird rotiert und in bufeb gepuffert und
 anschliessend zurueckkopiert. "p(lus)" ist mit dem Uhrzeigersinn
 v. vorne, rechts bzw. oben gesehen}

var  vorz:         tvorzeichen;

  procedure simplrot;
  {dreht ebene um 90 Grad plus - di. gegen den Uhrzeigersinn}

  var   bufeb:        tebene;         {lokale Pufferebene}
        ix, iy:       tindex;

    procedure rwuerf(inwuerf: twuerfel; var outwuerf: twuerfel);
    {kippt Teilwuerfel um angegebene Achse gedreht}

    begin
      outwuerf := inwuerf;
      with outwuerf, param do {zwei Flaechen vertauschen}
        case achse of
          xachse: begin z := inwuerf.y; y := inwuerf.z end;
          yachse: begin x := inwuerf.z; z := inwuerf.x end;
          zachse: begin x := inwuerf.y; y := inwuerf.x end;
        end {case};
    end {rwuerf};

  begin {simplrot / Rotation 90 Grad im Uhrzeigersinnn}
    with param do begin     {Ebene rotieren}
    for iy := 0 to 2 do
      for ix := 0 to 2 do
        case achse of
          xachse: rwuerf (kubus [achsindex, ix, iy], bufeb [iy, 2 - ix]);
          yachse: rwuerf (kubus [ix, achsindex, iy], bufeb [iy, 2 - ix]);
          zachse: rwuerf (kubus [ix, iy, achsindex], bufeb [iy, 2 - ix]);
        end {case};
    for iy := 0 to 2 do     {Ebene in Kubus zurueckkopieren}
      for ix := 0 to 2 do
        case achse of
          xachse: kubus [achsindex, ix, iy] := bufeb [ix, iy];
          yachse: kubus [ix, achsindex, iy] := bufeb [ix, iy];
          zachse: kubus [ix, iy, achsindex] := bufeb [ix, iy];
        end {case};
    end {with param};
  end {simplrot};
```

(bitte umblättern)

Der verflixte Würfel

```
begin (ro)
  with param do begin
    if umkehr then case vorzeichen of
                     plus:  vorzeichen := minus;
                     minus: vorzeichen := plus;
                     otherwise; (In UCSD alle 'otherwise' weglassen!)
                   end (case);
    for vorz := plus to vorzeichen do simplrot;
  end (with);
end (ro);

procedure display (out: boolean); (Abwicklung des Wuerfels auf Bildschirm )
(bzw. Wuerfelzustand einlesen)
(Merke: Wenn das Programm in ein Nicht-UCSD-System uebertragen werden
soll, muss ggf. fuer die Standard-Prozedur gotoxy(x, y: integer) ein
Ersatz geschaffen werden. Diese Prozedur positioniert den Cursor auf
dem Terminal bei den Koordinaten x, y. (0, 0) entspricht der home-
Position.)

  procedure dispxy (x, y: integer; achs: tdimindex; fix: tindex);

  var ix, iy: integer;

    procedure getfarb;

    var       i:      integer;
              f:      char;
              fn:     string [3];
              icl, cl:        tfarbe;

    begin
      repeat gotoxy (x + ix * 4 + 1, y + iy);
             fn := '   ';
             for i := 1 to 3 do begin
               read (f); fn [i] := f;
             end;
             if fn = '   ' then begin gutinit; display(true); exit(display)
                                  end;(ggf. muss statt 'exit(name)' ein)
             cl := schwarz; ('goto'zum Ende der Prozedur 'name' stehen!)
             for icl := weiss to gruen do
               if farbtabelle [icl] = fn then cl := icl;
      until cl <> schwarz;
      case achs of
        xachse: case fix of
          0: kubus [fix, 2 - iy, ix].x := cl;
          2: kubus [fix, 2 - iy, 2 - ix].x := cl;
                end;
        yachse: case fix of
           0: kubus [ix, fix, 2 - iy].y := cl;
          2: kubus [ix, fix, iy].y := cl;
                end;
        zachse: case fix of
          0: kubus [ix, iy, fix].z := cl;
          2: kubus [ix, 2 - iy, fix].z := cl;
                end;
      end;
    end (getfarb);

  begin (dispxy)
    for iy := 0 to 2 do begin
```

```
          gotoxy (x, y + iy);
          for ix := 0 to 2 do
            if out then case achs of
    xachse: case fix of
      0: with kubus [fix, 2 - iy, ix] do write (farbtabelle [x]:4);
      2: with kubus [fix, 2 - iy, 2 - ix] do write (farbtabelle [x]:4);
            end;
    yachse: case fix of
      0: with kubus [ix, fix, 2 - iy] do write (farbtabelle [y]:4);
      2: with kubus [ix, fix, iy] do write (farbtabelle [y]:4);
            end;
    zachse: case fix of
      0: with kubus [ix, iy, fix] do write (farbtabelle [z]:4);
      2: with kubus [ix, 2 - iy, fix] do write (farbtabelle [z]:4);
            end;
                        end
          else  getfarb;
        end (for iy);
      end (dispxy);

    begin (display)
      if not out then begin initschwarz; display (true) end;
      dispxy (14, 2, yachse, 2); gotoxy (32, 3); write ('Hinten'); (gruen)
      dispxy (0, 6, xachse, 0);    (orange)
      dispxy (14, 6, zachse, 2);   (weiss)
      dispxy (28, 6, xachse, 2);   (rot)
      dispxy (14, 10, yachse, 0); gotoxy (32, 11); write ('Vorne'); (blau)
      dispxy (14, 14, zachse, 0); gotoxy (32, 15); write ('Unten'); (gelb)
    end (display);

(B A C K T R A C K I N G   P L U S  -  P R O G R A M M)
(Uses Simulator & globale Utilities ************************************)

  procedure diswrite (instr: tinstruktion); (schreibt Operation im Klartext)

  begin
    with instr do begin  .
      if umkehr then write (drucker, ' -'); write (drucker, ' r');
      case achse of
        xachse:   write (drucker, 'x'); yachse:   write (drucker, 'y');
        zachse:   write (drucker, 'z');
      end;
      write (drucker, achsindex);
      case vorzeichen of
        plus:     write (drucker, 'p'); minus:    write (drucker, 'm');
        hundert:  write (drucker, 'h');
      end;
    end (with instr);
  end (diswrite);

  procedure savelist;     (schreibt Kardfile auf Diskette)

  begin
    close (kardfile);
    rewrite (kardfile, clistname);        (alter File wird geloescht!!!)
    lesepoin := listbas;
    repeat kardfile↑ := lesepoin↑; put (kardfile);
           lesepoin := lesepoin↑.linep;
    until lesepoin = nil;                 (Ende der Liste)
    close (kardfile, lock);
  end (savelist);
```

(bitte umblättern)

Der verflixte Würfel

```
procedure getlist;      (liest Kardfile von Diskette)

begin
  reset (kardfile, clistname);
  if ioresult <> 0 then begin gotoxy(0, 20);
                         write ('Diskfehler'); exit (getlist)
                        end
  else begin
    neupoin := listbas; lesepoin := listbas;
    repeat neupoin^ := kardfile^; get (kardfile);
           lesepoin^.liner := neupoin; neupoin^.liner := nil;
           lesepoin := neupoin; new (neupoin);
    until eof (kardfile);
    close (kardfile);
  end (else);
end (getlist);

procedure listkard;     (druckt Kardliste aus)

var     listfil:        text;
        zeile, zz:      integer;

begin
  rewrite (listfil, printer);
  zeile := 2; lesepoin := listbas;       (Anfang der Liste)
  writeln (listfil, 'Cube 1.1 Kardinalitaetsliste'); writeln (listfil);
  repeat with lesepoin^ do begin
           write (listfil, 'Kard: ');
           for zz := 1 to 6 do write (listfil, zeil [zz]:4);
           write (listfil, ' ':4, 'Levels: ');
           for zz := 0 to 6 do if zz in levs then write (listfil, zz:2);
           writeln (listfil, '.'); zeile := zeile + 1;
           lesepoin := lesepoin^.liner;
           if zeile >= 66 then begin
             for zeile := zeile + 1 to 72 do writeln (listfil); zeile := 0;
           end;
         end (with lesepoin);
  until lesepoin = nil;
  for zeile := zeile + 1 to 72 do writeln (listfil);
  close (listfil);
end (listkard);

procedure test(var l: tlevelset);
(intelligente Testroutine: Diskriminationsgrad 1 - 6 **********************)

var     gef:            boolean;
        zz, ix, iy:     integer;
        tf:             tfarbe;

begin (test) write ('.'); (M E N G E N S A M M L E R)
  for zz := 1 to 6 do farbt [zz] := [];     (Farbmengen = leer)
  for iy := 0 to 2 do
    for ix := 0 to 2 do begin
      (sammle fuer jede Seite die Menge der Farben)
      farbt [1] := farbt [1] + [kubus [ix, iy, 2].z];
      farbt [2] := farbt [2] + [kubus [ix, iy, 0].z];
      farbt [3] := farbt [3] + [kubus [0, ix, iy].x];
      farbt [4] := farbt [4] + [kubus [2, ix, iy].x];
      farbt [5] := farbt [5] + [kubus [ix, 0, iy].y];
      farbt [6] := farbt [6] + [kubus [ix, 2, iy].y];
```

```
      {Jede Seite bildet eine Farbmenge}
    end {for};
    for zz := 1 to 6 do ta [zz] := 0;
    {ermittle die Kardinalitaetsverteilung}
    for zz := 1 to 6 do begin  {fuer alle 6 Farbmengen}
      t := 0;
      for tf := weiss to gruen do
        if tf in farbt [zz]  then t := t + 1;
      ta [t] := ta [t] + 1; {erhoehe den Zaehler fuer eine Kardinalitaet}
    end {for zz};
    lesepoin := listbas;  {Anfang der Liste einstellen}
    l := []; gef := false;        {leere Menge}
    repeat if ta = lesepoint.zeil then begin
             l := lesepoint.levs; gef := true
           end;
           lesepoin := lesepoint.linep;   {naechste Zeile}
    until (lesepoin = nil) or gef;
    for zz := 0 to 6 do if zz in l then write (zz:2);
    write (': '); for zz := 1 to 6 do write (ta [zz], chr(clendline));
  end {test};

  procedure buildlist (n: integer);

  begin {buildlist - erzeugt Kardinalitaets - Liste}
    lesepoin := listbas;
    while (lesepoint.zeil <> ta) and (lesepoint.linep <> nil) do
      lesepoin := lesepoint.linep;
    if lesepoint.zeil = ta then lesepoint.levs := lesepoint.levs + [n]
          {fuege aktuellen Level zur Menge der Level hinzu}
    else begin    {neue Verteilungs-Zeile anfuegen}
      new (neupoin);
      with neupoint do begin
        linep := nil; zeil := ta; levs := [n]; {mit 'n' als erstem Element}
      end {with neupoint};
      lesepoint.linep := neupoin;         {an Liste anfuegen}
    end {else};
    if (versuch - (versuch div  500) * 500 = 0) then savelist;
       {Sicherheitskopie des Files}
  end {buildlist};

  procedure erfolg (n: integer);

  var         zz:     integer;

  begin clearscreen; prompt('Cube - Backtracking 1.1:');
        gotoxy (0, 20); write('Problem geloest im ', n, 'ten Zug nach ',
                        versuch, ' Versuchen');
        display (true);
        for zz := 1 to n do begin
          gotoxy (45, zz mod 21 + 2); diswrite (protlist [zz]);
        end {for}; fertig := true;
  end {erfolg};

  procedure backtrack (n: integer; donot: tinstruktion); {******************}

  var   testinstr:      tinstruktion;
        achsz:          tdimindex;
        achsindz:       tindex;
        vorzz:          tvorzeichen;

    procedure protokoll;
```

(bitte umblättern)

Der verflixte Würfel

```
  begin prompt('Backtracking Versuch: '); write (versuch, ' / ', maxlevel);
        gotoxy (45, n mod 21 + 2);
        if n mod 21 = 0 then begin clearscreen; display (true) end;
        write (n:4); diswrite (testinstr);
        protlist [n] := testinstr;
  end {protokoll};

begin {backtrack}
  gotoxy(0,23); write (memavail); {Wieviel Memory noch fuer Stack?}
  with testinstr do begin
    n := n + 1;          {Verschachtelungs-Tiefe}
    for achsz := xachse to zachse do
      for achsindz := 0 to 2 do
        for vorzz := plus to minus do
        if not ((achsz = donot.achse) and (achsindz = donot.achsindex))
           or (n = 1) then begin {vermeide redundante Versuche}
          versuch := versuch + 1;
          achse := achsz; achsindex := achsindz; vorzeichen := vorzz;
          umkehr := false;
          ro (testinstr); protokoll; {neue Operation - pruefe Ergebnis!}
          test (level);
          if senkard then begin
            if n < maxlevel then begin display (true); backtrack (n, testinstr)
                                 end
                            else buildlist (n)
          end
          else begin
            if 0 in level then begin erfolg (n);
                                     exit (backtrack)
                               end
             else if (maxlevel - n >= maxlev) then begin display (true);
                                                   backtrack (n, testinstr)
                                                 end
            {zu weit von Maxlevel entfernt, um ein Urteil abzugeben}
            else if maxlevel - n in level then backtrack (n, testinstr);
              {Das Ziel sollte erreichbar sein}
          end {else senkard};
          if fertig then exit (backtrack);
          umkehr := true; ro (testinstr);     {Op. ruecks. machen}
            {Wenn er zurueckgekommen ist, probiere etwas anderes}
        end {for};
  end {with testinstr};
end {backtrack};

procedure kardlist;     {erzeugt Liste der Kardinalitaetsverteilung}

var     max:    integer;

begin {kardlist}
  lesepoin := listbas; gutinit; display (true); startlev := 1; maxlev := 1;
  versuch := 0;
  repeat
    prompt('Generiere Kard-Liste: S(tartlevel, M(axlevel, R(un, E(nde ');
    writeln;
    write('Start: ', startlev, ' Maxlevel: ', maxlev, chr(clendline));
    repeat read (sw) until sw in ['s', 'S', 'm', 'M', 'r', 'R', 'e', 'E'];
    case sw of
      's', 'S': begin write(' Startlevel? '); read (startlev) end;
      'm', 'M': begin write(' Maxlevel? '); read (maxlev) end;
      'r', 'R': begin fertig := false;
```

```
                    for max := startlev to maxlev do
                    backtrack (0, dummy); savelist;
                end;
      'e', 'E': ;
    end;
  until sw in ['e', 'E'];
  sw := ' ';
end {kardlist};

procedure backals;

var     zz:     integer;

begin {backals}
  gotoxy (0, 1); write (chr(clendline));
  versuch  := 0; maxlev := 0;
  lesepoin := listbas;  {Anfang der Liste}
  repeat for zz := 1 to 6 do    {Finde groessten Schaetzlevel}
           if (zz in lesepoint.levs) and (zz > maxlev) then maxlev := zz;
         lesepoin := lesepoint.linep;
  until lesepoin = nil;
  maxlev := maxlev + 1; maxlevel := maxlev; {Unguenstigster Fall}
  test (level);
  for zz := 6 downto 0 do if zz in level then maxlevel := zz;
  {Abschaetzen der Ausgangssituation - mit welchem Level soll beg. werden}
  if maxlevel = 0 then maxlevel := 3; fertig := false;
  repeat backtrack (0, dummy);
         maxlevel := maxlevel + 1;
  until (maxlevel = cmaxn) or fertig;
end {backals};

begin {back **** H A U P T P R O G R A M M ******************************}
  new (listbas);        {Basis fuer Liste}
  with dummy do begin umkehr := false; achse := xachse; achsindex := 0;
                    vorzeichen := plus end; protlist [1] := dummy;
   with listbas^ do begin
    zeil [1] := 6; linep := nil;
     for zz := 2 to 6 do zeil [zz] := 0; levs := [0];
  end {with listbas};
  clearscreen; rewrite (drucker, konsol); getlist;
  farbinit; gutinit; display (true); versuch := 0;
  repeat
    prompt('Cube 1.1 Backtracking I(nput, D(isplay, R(un, E(nde ');
    writeln; write('K(ardlist, L(istkardfile , Versuche: ',
                   versuch, chr(clendline));
    repeat read (sw) until sw in ['I', 'i', 'D', 'd', 'R', 'r', 'E', 'e',
                                  'K', 'k', 'L', 'l'];
    case sw of
      'I', 'i': begin clearscreen; display (false) end;
       'D', 'd': begin clearscreen; display (true) end;
      'R', 'r': begin genkard := false; backals end;
       'K', 'k': begin genkard := true; kardlist end;
      'L', 'l': listkard;
       'E', 'e': ;
    end {case};
  until sw in ['E', 'e'];
end.
```

Jan Ropers

Mühleprogramm für alle Basic-Rechner (32K)

Beschreibung

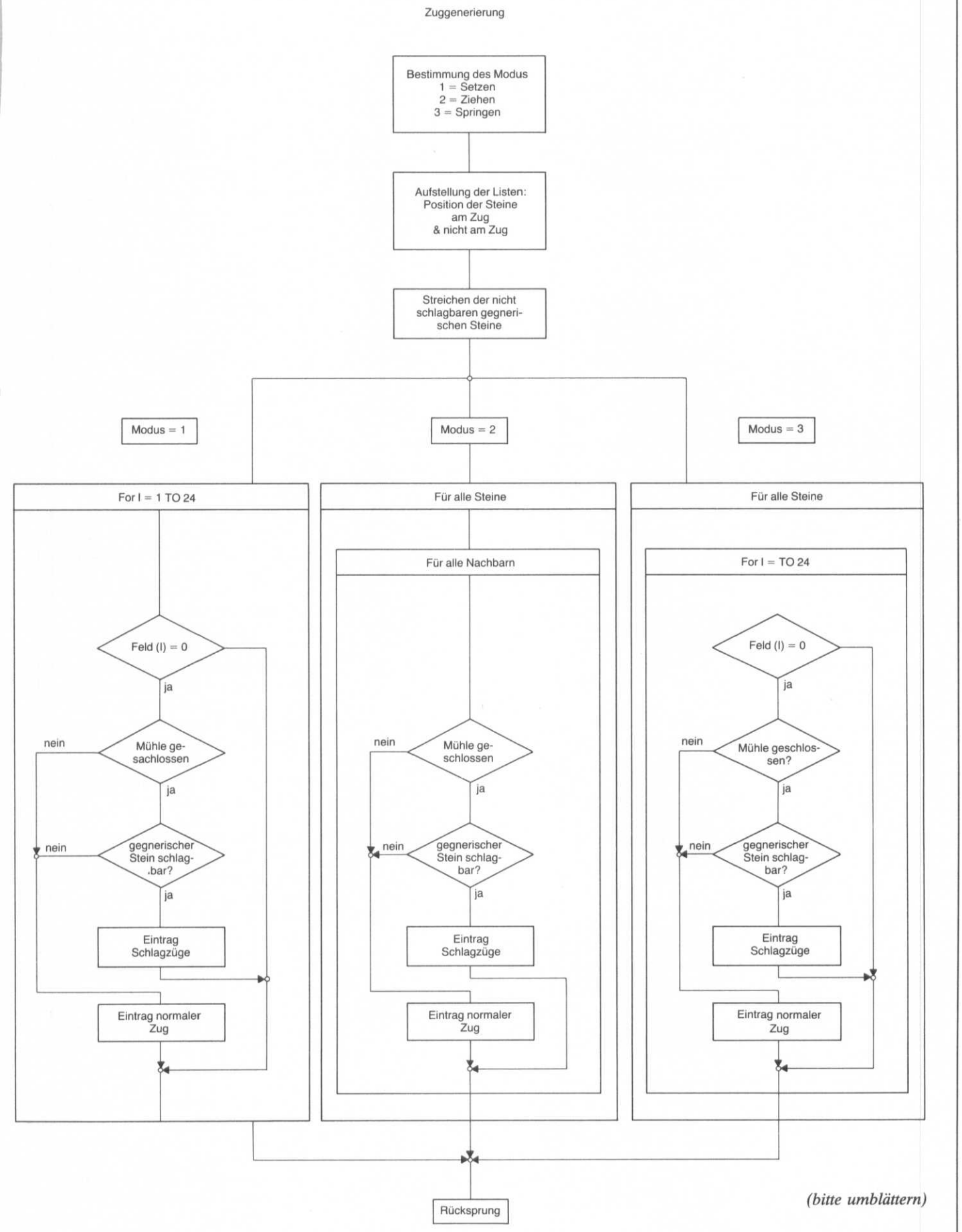
Da das Mühleprogramm in Basic geschrieben ist, und ich bei der Anordnung der Unterprogramme nicht auf Geschwindigkeit geachtet habe, benötigt es für eine Bewertung etwa 3 Sekunden. In der schwächsten Spielstufe entspricht dies einer Zugzeit von 20 Sekunden bis zu etwas über einer Minute, je nach Zahl der Zugmöglichkeiten. Im Mittelspiel beträgt die Zeit etwa 30 Sekunden. Bedingt durch die hohe Zahl an Zugmöglichkeiten im Endspiel (beim Springen) kann die Zugzeit für die letzten Züge bis zu 2,5 Minuten betragen. Die mittlere Zugzeit in der zweiten Spielstärke beträgt zwischen 2 und 3 Minuten. Alle höheren Spielstärken benötigen erheblich mehr Rechenzeit und sind daher außer für Tests indiskutabel. Da in das Programm aber im Gegensatz zum Kalah beide gebräuchlichen Vorsortierungsalgorithmen eingebaut sind, sollte man es sich trotzdem ansehen.

Der Programmaufbau entspricht dem üblichen Muster: Es gibt zunächst eine Initialisierung und im Anschluß daran wird abwechselnd der Spielerzug abgefragt bzw. der Computerzug bestimmt. Der „beste" Computerzug wird auf die folgende Weise bestimmt: Zunächst werden alle für den Computer erlaubten Züge erstellt. Dann wird ein Zug ausgeführt und der Computer bewertet die entstandene Stellung auf dem „Brett". Es wird also nicht ein Zug, sondern die daraus resultierende Stellung auf dem Spielfeld beurteilt. Nach dieser Bewertung wird der Zug wieder rückgängig gemacht und die erhaltene Bewertung wird jetzt dem Zug zugeordnet (da durch ihn ja die bewertete Stellung erreicht wurde). Dies Verfahren, Zug ausführen, entstandene Stellung bewerten und Zug wieder zurücknehmen, wird jetzt für alle anderen für den Computer erlaubten Züge ausgeführt und die entsprechenden Bewertungen den Zügen zugeordnet. Dann werden die Züge entsprechend den Bewertungen sortiert, die besten nach vorn und die schlechten nach hinten. Wenn diese Vorsortierung abgeschlossen ist, wird der erste, jetzt also der zunächst beste Computerzug ausgeführt und der Computer bestimmt die möglichen Spielerzüge. Wenn die Solltiefe bereits einmal erreicht wurde oder gerade erreicht ist wird die sogenannte Killerheuristik benutzt, um den wahrscheinlich besten Zug zuerst auszuführen. Ist die Solltiefe noch nicht erreicht worden, wird wie beim ersten Zug nach Bewertung sortiert. Nachdem jetzt mit einiger Wahrscheinlichkeit ein recht guter Antwortzug an erster Stelle der Liste steht, wird der erste Zug ausgeführt und der Computer erstellt jetzt wieder alle nun für ihn möglichen Züge und verfährt mit ihnen genauso wie mit den Spielerzügen. Dies Verfahren wird solange fortgesetzt, bis die Solltiefe erreicht ist, also bis die Zahl der *ausgeführten* Computer- und Spielerzüge mit einer vorgegebenen Zahl übereinstimmt. Auch jetzt werden die Züge nacheinander ausgeführt und die entstehenden Stellungen bewertet. Ist die einem Zug zugeordnete Bewertung jetzt besser als alle vorherigen (seit dem letzten Erreichen dieser Tiefe), so wird der Stellung vor dem Ausführen dieses Zuges der Wert zugeordnet und der Zug selbst als vermutlich guter Antwortzug auch für andere Stellungen gemerkt. Dieser Zug wird im weiteren Killerzug genannt. Nachdem alle Züge ausgeführt wurden und so der nach „Meinung" des Computers beste Antwortzug ermittelt wurde, wird der Zug eine Ebene höher rückgängig gemacht und ihm der Wert der Stellung zugeordnet. Dann wird in dieser höheren Ebene der nächste Zug ausgeführt, es wird wieder der beste Antwortzug bestimmt usw.

Auf diese Weise erhält man eine korrigierte Bewertung der möglichen Computerzüge. Wenn aber wirklich alles so ausgeführt werden würde, wie bisher beschrieben, würden alle möglichen Endstellungen erreicht werden und eine Vorsortierung der Züge wäre sinnlos.

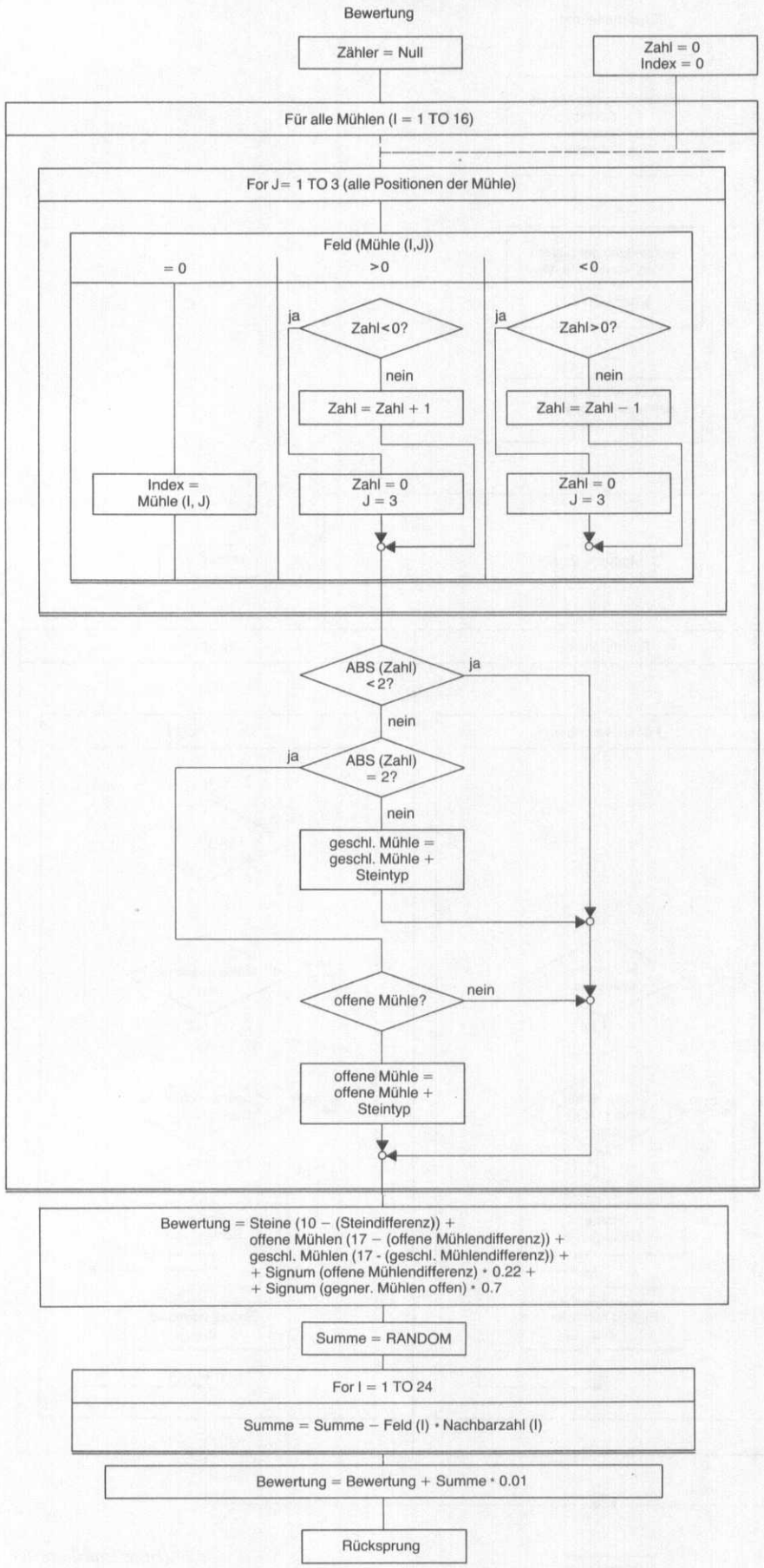
Der Grund für diese Vorsortierung ist folgender: Angenommen, der Computer soll sich lediglich Zug und Gegenzug ansehen. Außerdem sei der erste Zug und alle möglichen Gegenzüge bereits ausgeführt worden. Dann hat der erste Computerzug bereits seine endgültige Bewertung erhalten. Jetzt wird der zweite Computerzug ausgeführt, die möglichen Gegenzüge des Spielers werden erzeugt und der vorher beste Gegenzug wird, wenn er noch möglich ist, nach vorn sortiert. Dann wird der erste Zug ausgeführt, die Stellung bewertet, der Zug wieder zurückgenommen und die Stellung nach dem Computerzug erhält ihre erste vorläufige Bewertung. Wenn diese jetzt bereits schlechter ist als die endgültige Bewertung des ersten Computerzuges, brauchen die übrigen Gegenzüge gar nicht mehr ausgeführt und die Stellungen bewertet werden, da der Spieler den für ihn besten und damit den für den Computer schlechtesten Zug wählen wird. Damit wird der Wert der Stellung höchstens noch verschlechtert, nicht aber verbessert. Es ist in diesem Fall also bereits nach dem ersten Gegenzug klar, daß der zweite Computerzug nicht besser ist als der erste. Daher werden die restlichen Gegenzüge gar nicht ausgeführt. Auf diese Weise kann die Zahl der bewerteten Stellungen erheblich reduziert werden. Wenn n die Zahl der normalerweise zu bewertenden Stellungen ist, so reduziert sie sich für gerade Tiefen auf minimal $2\sqrt{n} - 1$. Im Fall von ungeraden Tiefen ist dies nicht ganz so günstig. Der Aufwand für die Sortierung lohnt sich aber auf jeden Fall, besonders wenn der Aufwand wie im Fall der Killerheuristik so klein ist (weit weniger Zeit als eine Bewertung je Sortierung).

Nun zur (kurzen) Beschreibung der einzelnen Programmteile. Im Programm werden mehrere Indexfelder benötigt. Dies sind zum Beispiel die Nachbarn



(bitte umblättern)

Mühleprogramm für alle Basic-Rechner (32K)



eines Steines, die Anzahl der Nachbarn, die Mühlen und ähnliches. Diese werden im Bereich 1 bis 990 und 4000 bis 4190 besetzt. Außerdem wird hier abgefragt, wer beginnen soll und wieviel Halbzüge der Computer verschachteln soll. Der Spielerzug wird im Bereich 7000 bis 7990 abgefragt. Hierzu werden zunächst alle erlaubten Spielerzüge erzeugt und anschließend wird der Zug eingelesen. Ist der unter den erlaubten, so wird er ausgeführt, ansonsten ein neuer eingelesen. Der Rahmen zur Bestimmung des besten Computerzuges ist im Bereich 6000 bis 6410. Wenn es sich nicht um den ersten Computerzug handelt und der Computer mehr als eine Zugmöglichkeit hat, wird der Bereich 6500 bis 6850 angesprungen, um den besten Zug für den Computer zu ermitteln. Sonst wird der einzig mögliche Zug sofort ausgeführt oder der Zug über Random bestimmt.

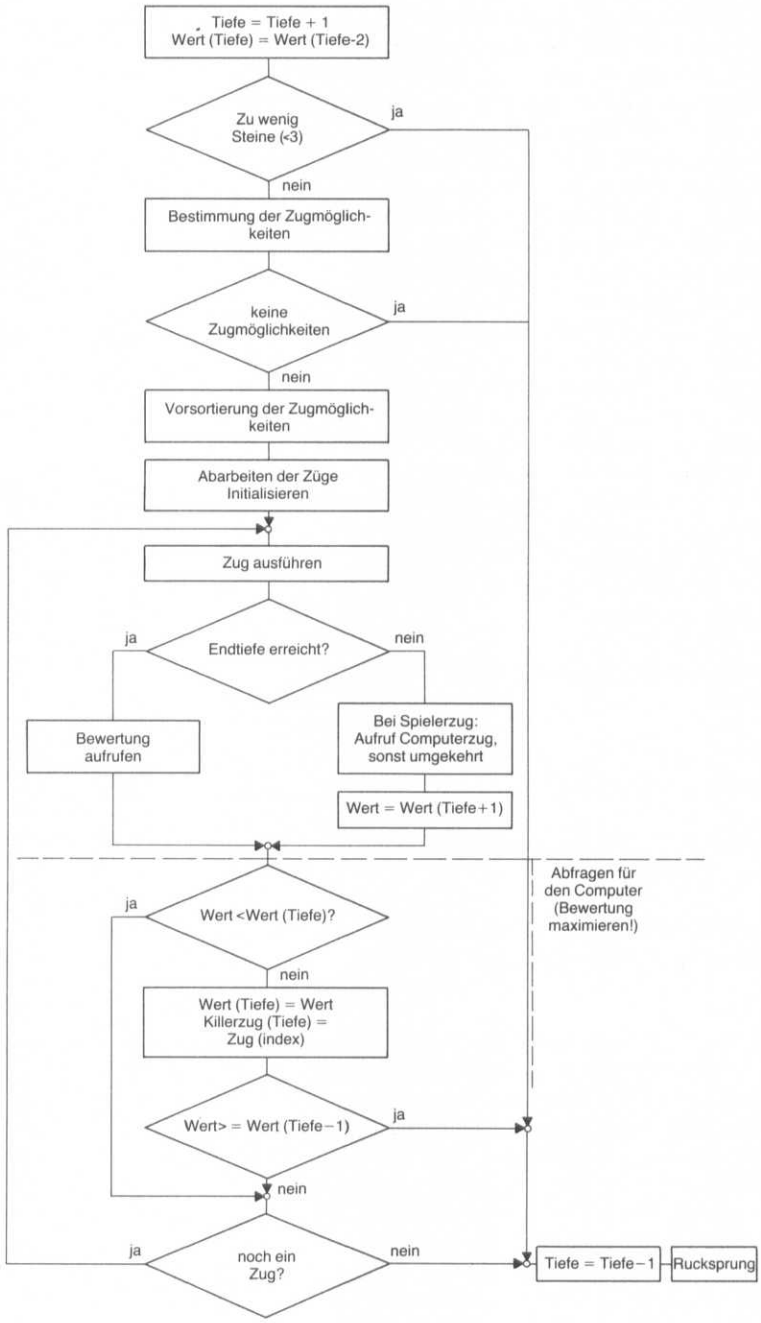
Im Bereich 6500 bis 6850 wird wie bereits erwähnt, der beste Computerzug bestimmt. Dieser Programmteil wird nur für den ersten Halbzug benutzt, da bei diesem keine Alpha-Beta-Abschneidungen vorkommen. Da dies Unterprogramm zur Bestimmung des besten Zuges nun einmal (je Zug) angesprungen wird, enthält es zunächst einmal die Initialisierung des Bewertungsfeldes und zweier Merkvariablen (P9 und M9). P9 enthält zum Schluß den Index des besten Computerzuges, M9 dient zur Unterscheidung, ob richtig vorsortiert werden soll (mit Bewertung), oder ob die Killerheuristik benutzt werden soll.

Nach der Initialisierung werden alle Züge der Reihe nach ausgeführt, der entstandenen Stellung und damit auch dem Zug ein Wert zugeordnet, sei es durch weitere Verschachtelung (Gegenzüge des Spielers) oder durch direkte Bewertung, und der Zug wieder zurückgenommen. Ist ein Zug besser als die vorhergehenden, so wird der Index des Zuges auf P9 gespeichert und der zugehörige Wert im Bewertungsfeld (zum Vergleich bei den nächsten Zügen und für Abschneidungen).

Wenn mehr als nur der Computerzug ausgeführt werden soll, wenn also mindestens Tiefe 2 gespielt wird, wird bei Erreichen einer geraden Tiefe (also wenn der Spieler am Zug ist), die Spielerverschachtelung im Bereich 5000 bis

5330, bei ungerader Tiefe die Computerverschachtelung im Bereich 5500 bis 5830 aufgerufen. Ab Tiefe 4 kommt es hier also zu Rekursionen. Daher müssen die Laufindizes und die Grenzen auf Feldern stehen, damit es nicht zu einem Durcheinander kommt.
Da beide Verschachtelungen bis auf die nötigen Umkehrungen in der Logik identisch sind, beschreibe ich im folgenden nur die Spielerverschachtelung. Diese beginnt mit dem Hochzählen der momentanen Verschachtelungstiefe und der Initialisierung der Bewertung für die momentane Stellung. Hierfür wird die bereits vorhandene Bewertung „zwei Stufen höher" gewählt. Wenn sich die Bewertung nämlich nicht verbessert, wird nach der Rückkehr in die nächst höhere Stufe dort festgestellt, daß noch eine Stufe höher dieser ganze Zweig ohnehin nicht mehr berücksichtigt wird. Eine Initialisierung mit einem von der Bewertung nicht erreichbaren Wert ist nicht sinnvoll, da dann eine Stufe tiefer beim ersten Zug noch keine Abschneidungen vorgenommen werden können, obwohl sie möglich sind.

Im Anschluß an diese Vorbesetzung werden die möglichen Züge erzeugt und wenn nötig (falls es mehr als einen gibt) „vorsortiert". Jetzt werden die Züge der Reihe nach ausgeführt, der entstandenen Stellung und damit auch dem Zug ein Wert zugeordnet und der Zug wieder zurückgenommen. Ist dieser Zug besser oder wenigstens genausogut wie die Vorbesetzung oder einer der vorherigen Züge, so wird der Wert der Stellung upgedatet und getestet, ob sie eventuell schon zum Abschneiden führt. Wenn dies der Fall ist, wird das Unterprogramm sofort beendet, ansonsten werden die anderen noch möglichen Züge ausgeführt.



Nun zu den beiden wichtigsten Unterprogrammen: die Bewertung und die Zuggenerierung. Hierzu zunächst einmal eine kurze Erklärung der Datenstruktur. Das Spielfeld ist auf einem Skalar der Länge 24 gespeichert. Das hat zur Folge, daß man Listen führen muß, auf welche Felder man von einem bestimmten Feld aus ziehen kann, welche Positionen eine Mühle ergeben usw. Allerdings wäre das auch im Fall eines 7 mal 7-Feldes notwendig, bedingt durch den Aufbau eines Mühlefelds. Eine Möglichkeit, die Mühlen einfach aus dem Feld heraus zu bestimmen, würde zwar ein 8 mal 3-Feld bieten, aber dann müßte man bei der Zuggenerierung mehr Sonderfälle betrachten. Auch bei offenen Mühlen gäbe es wahrscheinlich mehr zu programmieren. Die einzelnen Indexfelder enthalten folgende Information:

N(24, 4) enthält zu jedem Feld die möglichen Nachbarn
Z(24) enthält zu jedem Feld die Zahl der Nachbarn M(16, 3) enthält die 16 möglichen Mühlen
X(24, 2) enthält die beiden Positionen, die zusammen mit dem ersten Index eine Mühle bilden.
Y(24, 2) enthält die Position wie X, nur die der anderen möglichen Mühle.
G(4, 255) dient als Speicher für die Zugmöglichkeiten
T(2, 11) enthält die Killerzüge
F(24) ist das Spielfeld

(bitte umblättern)

Mühleprogramm für alle Basic-Rechner (32K)

Nun zur Bewertung. Sie belegt den Bereich von 4200 bis 4790. Es werden zunächst die offenen und geschlossenen Mühlen beider Seiten gezählt und voneinander subtrahiert. Außerdem wird gestestet, ob die Partei, die als nächstes am Zug ist, eine Mühle offen hat. Dies geschieht in den Zeilen 4240 bis 4620. Mo enthält danach die Zahl der offenen Mühlen des Rechners abzüglich der offenen des Spielers, M1 den entsprechenden Wert für geschlossene Mühlen. Wenn M2 gleich Null ist, hat die Partei, die als nächstes am Zug ist, keine Mühle offen, im anderen Fall ist M2 plus oder minus eins, je nachdem, ob der Spieler oder der Computer als nächstes am Zug ist. Diese Werte ergeben zusammen mit der jeweiligen Zahl der noch im Spiel befindlichen Steine eine vorläufige Bewertung. Diese ergibt sich genau aus

Tabellenwert Computersteine — Spielersteine

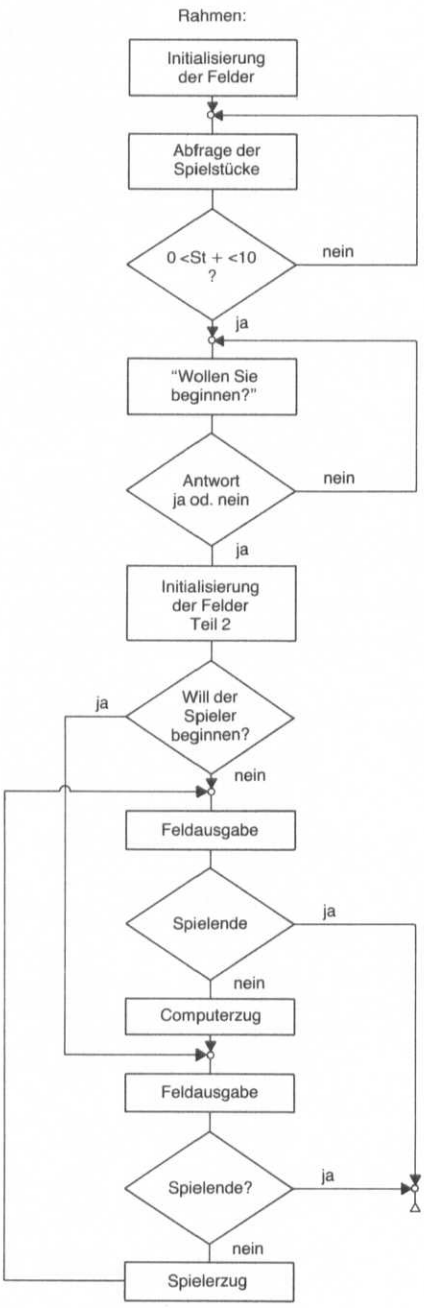
Tabellenwert Computermühlen — Spielermühlen (offen)

Tabellenwert Computermühlen — Spielermühlen (zu)

Bonus für Mühlenüberschuß 0.22
Bonus für offene Mühle und am Zug 0.7

Anschließend wird noch die Position der einzelnen Steine bewertet ohne Rücksicht auf die anderen. Dies geschieht, indem die Zahl der Nachbarn als Bewertung aufsummiert wird und anschließend der übrigen Bewertung angeglichen wird. Die Fortranversion des Spiels hatte zusätzlich noch nachgesehen, wieviel Steine so stehen, daß durch Hinzufügen eines weiteren zwei Mühlen gleichzeitig aufgemacht werden. Dies erfordert aber eine Kaskade von IF-Abfragen über mehr als 200 Zeilen (was für dieses Basicprogramm ein übertriebener Aufwand wäre) oder eine sehr langsame dreifache Schleife, wobei die benötigte Rechenzeit in keinem vernünftigen Verhältnis mehr zur gewonnenen Information steht.
In die Bewertung geht ein bißchen Random mit ein, um auszuschließen, daß der Computer in der Eröffnung immer dieselben Züge macht.

Zum Abschluß folgt jetzt die kurze Beschreibung der Zuggenerierung. Es



werden zunächst für beide Parteien Listen aufgestellt, wo sich deren Steine befinden. Dann werden die Steine der Partei, die *nicht* am Zug ist, daraufhin überprüft, ob sie eine geschlossene Mühle bilden. Wenn dies der Fall ist, werden die entsprechenden Steine entfernt. Kann jetzt die Partei, die am Zug ist, eine Mühle schließen, ist sofort bekannt, welche gegnerischen Steine entfernt werden können. *Achtung!* An dieser Stelle entspricht diese Version eventuell nicht den normalen Mühlere-geln. Es kann nämlich passieren, daß gar keine Steine entfernt werden dürfen, unabhängig davon, wieviel Steine beide Parteien haben. Wenn die beiden Listen stehen, wird in die drei Zweige Setzen, Ziehen und Springen verzweigt, in Abhängigkeit von der Zahl der Steine auf dem Brett und außerhalb. Dies sind aber einfache Schleifen, so daß ich auf sie nicht näher eingehe.

P.S.: Die Funktion USER(8) in den Zeilen 7000 bis 8000 liefert den ASCII-Wert eines auf der Tastatur eingegebenen Zeichens (sie wartet, bis ein Zeichen kommt und wandelt dann). Diese unkonventionelle Art der Zugabfrage ist in meinem Spezialbasic nötig, da mir keine Strings zur Verfügung stehen und Zeichen eingelesen werden müssen.

```
0010 REM
0020 REM Muehle Version 1.0
0030 REM 25.1.1981 by Jan Ropers
0040 REM
0050 REM Definition der Felder
0060 REM
0070 DIM F( 24 ) , K( 9 ) , L( 9 )
0080 DIM Z( 24 ) , N( 24 , 4 )
0090 DIM M( 16 , 3 ) , X( 24 , 2 ) , Y( 24 , 2 )
0100 DIM I( 7 , 7 )
0110 DIM A( 24 ) , O( 24 )
0120 DIM G( 4 , 255 ) , T( 2 , 11 )
0130 DIM W( 11 ) , P( 11 ) , Q( 11 )
0140 DIM B( 19 ) , C( 33 ) , D( 33 )
0150 REM
0160 REM Initialisierung
0170 REM
0180  I = 0
0190  J = 0
0200  K = 0
0210  L = 0
0220  F = 0
0230  P = 0
0240 REM
0250 FOR I = 1 TO 24
0260  F( I ) = 0
0270 NEXT I
0280 REM
0290 RESTORE
0300 REM
0310 FOR I = 1 TO 24
0320 READ Z( I )
0330 FOR J = 1 TO 4
0340 READ N( I , J )
0350 NEXT J
0360 NEXT I
0370 REM
0380 FOR I = 1 TO 16
0390 READ J
0400 FOR J = 1 TO 3
0410 READ M( I , J )
0420 NEXT J
0430 NEXT I
0440 REM
0450 FOR I = 1 TO 7
0460 READ J
0470 FOR J = 1 TO 7
0480 READ I( I , J )
0490 IF I( I , J ) = 0 THEN 520
0500  A( I( I , J ) ) = J + 64
0510  O( I( I , J ) ) = I
0520 NEXT J
0530 NEXT I
0540 REM
0550  C = 0
0560  C0 = 9
0570  C9 = 9
0580  S = 0
0590  S0 = 9
0600  S9 = 9
0610 REM
0620 FOR I = 1 TO 32
0630 PRINT
0640 NEXT I
0650 REM
0660 PRINT " Welche Spielstaerke ( 1-9 ) ?" ;
0670 INPUT T0
0680 PRINT
0690  T0 = INT( T0 + 0.5 )
0700 IF T0 > 0 THEN IF T0 < 10 THEN 740
0710 PRINT " Diese Spielstaerke ist nicht moeglich"
0720 PRINT
0730 GOTO 660
0740  T0 = T0 + 2
0750 PRINT
0760 PRINT " Wollen Sie beginnen ( 1/0 ) ?" ;
0770 INPUT J
0780 PRINT
0790  J = INT( J + 0.5 )
0800 IF J < 0 THEN 750
0810 IF J > 1 THEN 750
0820 REM Initialisierung Teil 2
0830 FOR I = 1 TO 8
0840  X( M( I , 1 ) , 1 ) = M( I , 2 )
0850  X( M( I , 1 ) , 2 ) = M( I , 3 )
0860  X( M( I , 2 ) , 1 ) = M( I , 1 )
0870  X( M( I , 2 ) , 2 ) = M( I , 3 )
0880  X( M( I , 3 ) , 1 ) = M( I , 1 )
0890  X( M( I , 3 ) , 2 ) = M( I , 2 )
0900 NEXT I
0910 FOR I = 9 TO 16
0920  Y( M( I , 1 ) , 1 ) = M( I , 2 )
0930  Y( M( I , 1 ) , 2 ) = M( I , 3 )
0940  Y( M( I , 2 ) , 1 ) = M( I , 1 )
0950  Y( M( I , 2 ) , 2 ) = M( I , 3 )
0960  Y( M( I , 3 ) , 1 ) = M( I , 1 )
0970  Y( M( I , 3 ) , 2 ) = M( I , 2 )
0980 NEXT I
0990 GOTO 4000
1000 REM
1010 REM Erstellung aller Zuege fuer F
1020 REM
1030  F1 = -F
1040 ON SGN( F ) + 2 GOTO 1100 , 1050 , 1200
1050 REM Falsche Versorgung
1060 STOP
1100 REM Zuege fuer den Rechner
1110  M0 = 2 - SGN( C9 )
1120 IF C0 <= 3 THEN M0 = 3
1130 GOTO 1300
1200 REM Zuege fuer den Spieler
1210  M0 = 2 - SGN( S9 )
1220 IF S0 <= 3 THEN M0 = 3
1300 REM Position der Steine
1310  K = 0
1320  L = 0
1330 FOR I = 1 TO 24
1340 ON F * F( I ) + 2 GOTO 1350 , 1400 , 1380
1350  K = K + 1
1360  K( K ) = I
1370 GOTO 1400
1380  L = L + 1
1390  L( L ) = I
1400 NEXT I
1410 REM Loeschen der nicht schlagbaren gegnerischen Steine
1420 FOR I = 1 TO 16
1430 IF F( M( I , 1 ) ) <> F1 THEN 1600
1440 IF F( M( I , 2 ) ) <> F1 THEN 1600
1450 IF F( M( I , 3 ) ) <> F1 THEN 1600
1460  M = 1
1470  N = 1
1480 ON SGN( K( N ) - M( I , M ) ) + 2 GOTO 1490 , 1520 , 1580
1490  N = N + 1
1500 IF N <= K THEN 1480
1510 GOTO 1600
1520  K = K - 1
1530 IF K < N THEN 1600
1540 FOR O = N TO K
1550  K( O ) = K( O + 1 )
1560 NEXT O
1580  M = M + 1
1590 IF M <= 3 THEN 1480
1600 NEXT I
1610 REM Verzweigung in die drei Zweige
1620  G = P( T - 1 )
1630 ON M0 GOTO 1700 , 2000 , 2300
1700 REM Setzen der Steine
1710 FOR I = 1 TO 24
1720 IF F( I ) <> 0 THEN 1890
1730 IF K = 0 THEN 1850
1740 IF F <> F( X( I , 1 ) ) THEN 1760
1750 IF F = F( X( I , 2 ) ) THEN 1780
1760 IF F <> F( Y( I , 1 ) ) THEN 1850
1770 IF F <> F( Y( I , 2 ) ) THEN 1850
1780 FOR J = 1 TO K
1790  G = G + 1
1800  G( 1 , G ) = 0
1810  G( 2 , G ) = I
1820  G( 3 , G ) = K( J )
1830 NEXT J
1840 GOTO 1890
1850  G = G + 1
1860  G( 1 , G ) = 0
1870  G( 2 , G ) = I
1880  G( 3 , G ) = 0
1890 NEXT I
1900  P( T ) = G
1910  G = G - P( T - 1 )
1920 RETURN
2000 REM Ziehen der Steine
2010 FOR I = 1 TO L
2020  P = L( I )
2030  F( P ) = 0
2040 FOR J = 1 TO Z( P )
2050  P1 = N( P , J )
2060 IF F( P1 ) <> 0 THEN 2230
2070 IF K = 0 THEN 2190
2080 IF F <> F( X( P1 , 1 ) ) THEN 2100
2090 IF F = F( X( P1 , 2 ) ) THEN 2120
2100 IF F <> F( Y( P1 , 1 ) ) THEN 2190
2110 IF F <> F( Y( P1 , 2 ) ) THEN 2190
2120 FOR J1 = 1 TO K
2130  G = G + 1
2140  G( 1 , G ) = P
2150  G( 2 , G ) = P1
2160  G( 3 , G ) = K( J1 )
2170 NEXT J1
2180 GOTO 2230
2190  G = G + 1
2200  G( 1 , G ) = P
2210  G( 2 , G ) = P1
2220  G( 3 , G ) = 0
2230 NEXT J
2240  F( P ) = F
2250 NEXT I
2260  P( T ) = G
2270  G = G - P( T - 1 )
2280 RETURN
```

(bitte umblättern)

Mühleprogramm für alle Basic-Rechner (32K)

```
2300 REM Springen der Steine
2310 FOR I = 1 TO L
2320  P = L( I )
2330 FOR J = 1 TO 24
2340 IF F( J ) <> 0 THEN 2540
2350 IF K = 0 THEN 2500
2360  F( P ) = 0
2370 IF F <> F( X( J , 1 ) ) THEN 2390
2380 IF F = F( X( J , 2 ) ) THEN 2410
2390 IF F <> F( Y( J , 1 ) ) THEN 2490
2400 IF F <> F( Y( J , 2 ) ) THEN 2490
2410 FOR J1 = 1 TO K
2420  G = G + 1
2430  G( 1 , G ) = P
2440  G( 2 , G ) = J
2450  G( 3 , G ) = K( J1 )
2460 NEXT J1
2470  F( P ) = F
2480 GOTO 2540
2490  F( P ) = F
2500  G = G + 1
2510  G( 1 , G ) = P
2520  G( 2 , G ) = J
2530  G( 3 , G ) = 0
2540 NEXT J
2550 NEXT I
2560  P( T ) = G
2570  G = G - P( T - 1 )
2580 RETURN
2600 REM
2610 REM Vorsortierung der Zuege
2620 REM
2630  V7 = P( T - 1 ) + 1
2640  B7 = P( T )
2650 REM
2655 IF V7 >= B7 THEN RETURN
2660 IF M9 = 1 THEN 2900
2665 IF T >= T0 THEN 2900
2670 REM
2680 FOR P = V7 TO B7
2690 ON F + 2 GOSUB 3000 , 9999 , 3500
2700 GOSUB 4200
2710  G( 4 , P ) = - W * F
2720 ON F + 2 GOSUB 3200 , 9999 , 3700
2730 NEXT P
2740 FOR I = V7 TO B7 - 1
2745  W = G( 4 , I )
2750  P = I
2755 FOR J = I + 1 TO B7
2760 IF G( 4 , J ) <= W THEN 2775
2765  W = G( 4 , J )
2770  P = J
2775 NEXT J
2780 IF P = I THEN 2835
2785  W = G( 1 , I )
2790  G( 1 , I ) = G( 1 , P )
2795  G( 1 , P ) = W
2800  W = G( 2 , I )
2805  G( 2 , I ) = G( 2 , P )
2810  G( 2 , P ) = W
2815  W = G( 3 , I )
2820  G( 3 , I ) = G( 3 , P )
2825  G( 3 , P ) = W
2830  G( 4 , P ) = G( 4 , I )
2835 NEXT I
2840 REM
2845  M9 = 0
2850 REM
2855 RETURN
2900 REM Killerzug nach vorn
2905  P0 = V7
2910  P = V7
2915 IF T( 2 , T ) <> G( 2 , P ) THEN 2975
2920 IF T( 1 , T ) <> G( 1 , P ) THEN 2975
2925  G( 1 , P ) = G( 1 , P0 )
2930  G( 2 , P ) = G( 2 , P0 )
2935  W = G( 3 , P )
2940  G( 3 , P ) = G( 3 , P0 )
2945  G( 1 , P0 ) = T( 1 , T )
2950  G( 2 , P0 ) = T( 2 , T )
2955  G( 3 , P0 ) = W
2960  P0 = P0 + 1
2965 REM
2970 IF W = 0 THEN RETURN
2975 REM
2980  P = P + 1
2985 IF P <= B7 THEN 2915
2990 REM
2995 RETURN
3000 REM
3010 REM Ausfuehren eines Computerzuges
3020 REM
3030 IF G( 1 , P ) = 0 THEN 3060
3040  F( G( 1 , P ) ) = 0
3050 GOTO 3080
3060  C9 = C9 - 1
3070  C = C + 1
3080  F( G( 2 , P ) ) = -1
3090 IF G( 3 , P ) = 0 THEN RETURN
3100  F( G( 3 , P ) ) = 0
3110  S = S - 1
3120  S0 = S0 - 1
3130 RETURN
3200 REM
3210 REM Zuruecknehmen eines Computerzuges
3220 REM
3230 IF G( 1 , P ) = 0 THEN 3260
3240  F( G( 1 , P ) ) = -1
3250 GOTO 3280
3260  C9 = C9 + 1
3270  C = C - 1
3280  F( G( 2 , P ) ) = 0
3290 IF G( 3 , P ) = 0 THEN RETURN
3300  F( G( 3 , P ) ) = 1
3310  S = S + 1
3320  S0 = S0 + 1
3330 RETURN
3500 REM
3510 REM Ausfuehren eines Spielerzuges
3520 REM
3530 IF G( 1 , P ) = 0 THEN 3560
3540  F( G( 1 , P ) ) = 0
3550 GOTO 3580
3560  S9 = S9 - 1
3570  S = S + 1
3580  F( G( 2 , P ) ) = 1
3590 IF G( 3 , P ) = 0 THEN RETURN
3600  F( G( 3 , P ) ) = 0
3610  C = C - 1
3620  C0 = C0 - 1
3630 RETURN
3700 REM
3710 REM Zuruecknehmen eines Spielerzuges
3720 REM
3730 IF G( 1 , P ) = 0 THEN 3760
3740  F( G( 1 , P ) ) = 1
3750 GOTO 3780
3760  S9 = S9 + 1
3770  S = S - 1
3780  F( G( 2 , P ) ) = 0
3790 IF G( 3 , P ) = 0 THEN RETURN
3800  F( G( 3 , P ) ) = -1
3810  C = C + 1
3820  C0 = C0 + 1
3830 RETURN
4000 REM
4010 REM Initialisierung der Bewertung
4020 REM
4030 FOR I = 1 TO 9
4040 READ B( I )
4050  B( 20 - I ) = - B( I )
4060 NEXT I
4070  B( 10 ) = 0
4080 FOR I = 1 TO 16
4090 READ C( I )
4100  C( 34 - I ) = - C( I )
4110 NEXT I
4120  C( 17 ) = 0
4130 FOR I = 1 TO 16
4140 READ D( I )
4150  D( 34 - I ) = - D( I )
4160 NEXT I
4170  D( 17 ) = 0
4180 REM Sprung in die Spielschleife
4190 ON J + 1 GOTO 6000 , 7000
4200 REM
4210 REM Bewertung ( fuer den Rechner )
4220 REM
4230  W = 0
4240 REM Zaehlen der Muehlen
4250  M0 = 0
4255  M1 = 0
4260  M2 = 0
4270 FOR I = 1 TO 16
4280  N = 0
4290  A = 0
4300 FOR J = 1 TO 3
4310 ON F( M( I , J ) ) + 2 GOTO 4320 , 4380 , 4350
4320 IF A > 0 THEN 4400
4330  A = A - 1
4340 GOTO 4420
4350 IF A < 0 THEN 4400
4360  A = A + 1
4370 GOTO 4420
4380  N = M( I , J )
4390 GOTO 4420
4400  J = 3
4410  A = 0
4420 NEXT J
4430 ON ABS( A ) + 1 GOTO 4620 , 4620 , 4440 , 4580
4440  A = SGN( A )
4450 FOR J = 1 TO Z( N )
4460  N1 = N( N , J )
4470 IF F( N1 ) <> A THEN 4560
4480 IF M( I , 1 ) = N1 THEN 4560
4490 IF M( I , 2 ) = N1 THEN 4560
4500 IF M( I , 3 ) = N1 THEN 4560
4510  J = 6
```

```
4520 ON A + 2 GOTO 4530 , 4560 , 4550
4530  M0 = M0 + 1
4535 IF F = 1 THEN M2 = 1
4540 GOTO 4560
4550  M0 = M0 - 1
4555 IF F = -1 THEN M2 = -1
4560 NEXT J
4562 IF J = 6 THEN 4620
4564 ON A + 2 GOTO 4566 , 4620 , 4574
4566 IF C9 = 0 THEN IF C0 <> 3 THEN 4620
4568  M0 = M0 + 1
4570 IF F = 1 THEN M2 = 1
4572 GOTO 4620
4574 IF S9 = 0 THEN IF S0 <> 3 THEN 4620
4576  M0 = M0 - 1
4578 IF F = -1 THEN M2 = -1
4579 GOTO 4620
4580 ON SGN( A ) + 2 GOTO 4590 , 4620 , 4610
4590  M1 = M1 + 1
4600 GOTO 4620
4610  M1 = M1 - 1
4620 NEXT I
4630 REM
4640 REM Fuer's erste Ende der Bewertung
4650 REM
4660  W = B( 10 - ( C0 - S0 ) ) + C( 17 - M0 ) + D( 17 - M1 )
4670  W = W + M2 * 0.7 + SGN( M0 ) * 0.22
4710  W0 = RND( 0 )
4720 FOR I = 1 TO 24
4730  W0 = W0 - F( I ) * Z( I )
4740 NEXT I
4750  W = W + W0 * 0.01
4760 REM
4770  M9 = 1
4780 REM
4790 RETURN
4800 REM
4810 REM Spieler hat verloren
4820 REM
4830 PRINT
4840 PRINT
4850 PRINT
4860 PRINT " Sie haben verloren."
4870 PRINT
4880 PRINT
4890 END
4900 REM
4910 REM Computer hat verloren
4920 REM
4930 PRINT
4940 PRINT
4950 PRINT
4960 PRINT " Sie haben gewonnen."
4970 PRINT
4980 PRINT
4990 END
5000 REM
5010 REM Verschachtelung Spieler
5020 REM
5030  T = T + 1
5040  W( T ) = W( T - 2 )
5045 REM
5050 IF S0 < 3 THEN 5290
5055 REM
5060  F = 1
5070 REM
5080 GOSUB 1000
5090 REM
5100 IF G = 0 THEN 5290
5110  Q( T ) = P( T - 1 ) + 1
5115 GOSUB 2600
5120 REM
5130  P = Q( T )
5140 REM
5150 GOSUB 3500
5160 IF T = T0 THEN 5210
5170 GOSUB 5500
5180  W = W( T + 1 )
5190  P = Q( T )
5200 GOTO 5220
5210 GOSUB 4200
5220 GOSUB 3700
5230 ON SGN( W - W( T ) ) + 2 GOTO 5235 , 5250 , 5260
5235  W( T ) = W
5240  T( 1 , T ) = G( 1 , P )
5245  T( 2 , T ) = G( 2 , P )
5250 IF W <= W( T - 1 ) THEN 5290
5260  P = P + 1
5270  Q( T ) = P
5280 IF P <= P( T ) THEN 5140
5290 REM
5300  F = -1
5310  T = T - 1
5320 REM
5330 RETURN
5500 REM
5510 REM Verschachtelung Computer ( 3 + )
5520 REM
5530  T = T + 1
5540  W( T ) = W( T - 2 )
5545 REM
5550 IF C0 < 3 THEN 5790
5555 REM
5560  F = -1
5570 REM
5580 GOSUB 1000
5590 REM
5600 IF G = 0 THEN 5790
5610  Q( T ) = P( T - 1 ) + 1
5615 IF T < T0 THEN GOSUB 2600
5620 REM
5630  P = Q( T )
5640 REM
5650 GOSUB 3000
5660 IF T = T0 THEN 5710
5670 GOSUB 5000
5680  W = W( T + 1 )
5690  P = Q( T )
5700 GOTO 5720
5710 GOSUB 4200
5720 GOSUB 3200
5730 ON SGN( W( T ) - W ) + 2 GOTO 5735 , 5750 , 5760
5735  W( T ) = W
5740  T( 1 , T ) = G( 1 , P )
5745  T( 2 , T ) = G( 2 , P )
5750 IF W >= W( T - 1 ) THEN 5790
5760  P = P + 1
5770  Q( T ) = P
5780 IF P <= P( T ) THEN 5640
5790 REM
5800  F = 1
5810  T = T - 1
5820 REM
5830 RETURN
6000 REM
6010 REM Computerzug
6020 REM
6030 GOSUB 8000
6040 REM
6050 IF C0 < 3 THEN 4900
6060  F = -1
6065  T = 3
6070 IF C9 + S9 > 2 THEN T = 4
6075 IF C0 = 3 THEN T = 4
6080 IF S0 = 3 THEN T = 4
6085  P( T - 1 ) = 0
6090 GOSUB 1000
6100 IF G = 0 THEN 4900
6110 REM
6120  P = 1
6130 IF G = 1 THEN 6310
6140 REM
6150 IF C9 < 9 THEN 6290
6155  P = INT(12.6+SGN(RND(0)-RND(0))*(1.5+6*INT(RND(0)+0.5)))
6160 IF S9 = 9 THEN 6310
6170 REM
6180  G = 0
6190 FOR I = 1 TO 24
6200 IF F( I ) <> 1 THEN 6270
6210 FOR J = 1 TO Z( I )
6220  G = G + 1
6230  G( 1 , G ) = 0
6240  G( 2 , G ) = N( I , J )
6250  G( 3 , G ) = 0
6260 NEXT J
6270 NEXT I
6280  P( T ) = G
6290 REM
6300 GOSUB 6500
6310 REM
6320 GOSUB 3000
6330 REM PRINT
6340 PRINT " Computerzug : " ;
6350 FOR I = 1 TO 3
6360  J = G( I , P )
6370 IF J > 0 THEN PRINT CHR$( A( J ) ) ; O( J ) ;
6380 NEXT I
6390 PRINT
6400 REM
6410 GOTO 7000
6500 REM
6510 REM Bestimmung des besten Computerzuges
6520 REM
6530  P9 = 0
6535  M9 = 0
6540  W( T - 1 ) =  1.0E20
6550  W(   T   ) = -1.0E20
6560  Q(   T   ) = P( T - 1 ) + 1
6565 IF T < T0 THEN GOSUB 2600
6570 REM
6580 REM und Jetzt die grosse Schleife
6590 REM
6595 IF T < T0 THEN PRINT " " ;
6600  P = Q( T )
6610 REM
6615 GOSUB 3000
6620 IF T >= T0 THEN 6670
```

(bitte umblättern)

Mühleprogramm für alle Basic-Rechner (32K)

```
6625 PRINT "*" ;
6630 GOSUB 5000
6640  W = W( T + 1 )
6650  P = Q( T )
6660 GOTO 6680
6670 GOSUB 4200
6680 GOSUB 3200
6690 IF W <= W( T ) THEN 6720
6700  W( T ) = W
6710  P9 = P
6720  P = P + 1
6730  Q( T ) = P
6740 IF P <= P( T ) THEN 6610
6750 REM
6760 REM Ende der grossen Schleife
6770 REM
6780  P = P9
6790 REM
6800 IF T >= T0 THEN RETURN
6810 REM
6820 PRINT
6830 PRINT
6840 REM
6850 RETURN
7000 REM
7010 REM Spielerzug
7020 REM
7030 GOSUB 8000
7040 REM
7050 IF S0 < 3 THEN 4800
7060  F = 1
7070  T = 2
7075  P( T - 1 ) = 0
7080 GOSUB 1000
7090 IF G = 0 THEN 4800
7100 REM
7110 REM Einlesen des Spielerzuges
7120 REM
7130 REM PRINT
7140 PRINT " Spielerzug : " ;
7150  V0 = USER( 8 )
7160 IF V0 < 65 THEN 7150
7170 IF V0 > 71 THEN 7150
7180 PRINT CHR$( V0 ) ;
7190  V1 = USER( 8 )
7200 IF V1 < 49 THEN 7190
7210 IF V1 > 55 THEN 7190
7220 PRINT CHR$( V1 ) ;
7230  N0 = USER( 8 )
7240 IF N0 = 13 THEN 7750
7250 IF N0 < 65 THEN 7230
7260 IF N0 > 71 THEN 7230
7270 PRINT CHR$( N0 ) ;
7280  N1 = USER( 8 )
7290 IF N1 < 49 THEN 7280
7300 IF N1 > 55 THEN 7280
7310 PRINT CHR$( N1 ) ;
7320  E0 = USER( 8 )
7330 IF E0 = 13 THEN 7530
7340 IF E0 < 65 THEN 7320
7350 IF E0 > 71 THEN 7320
7360 PRINT CHR$( E0 ) ;
7370  E1 = USER( 8 )
7380 IF E1 < 49 THEN 7370
7390 IF E1 > 55 THEN 7370
7400 PRINT CHR$( E1 ) ;
7410 REM
7420 REM Ziehen und Entfernen eines Steines
7430 REM
7440  V = I( V1 - 48 , V0 - 64 )
7450  N = I( N1 - 48 , N0 - 64 )
7460  E = I( E1 - 48 , E0 - 64 )
7470 REM
7480 IF V = 0 THEN 7970
7490 IF N = 0 THEN 7970
7500 IF E = 0 THEN 7970
7510 REM
7520 GOTO 7830
7530 REM
7540 REM Ziehen oder Setzen und Entfernen eines Steines
7550 REM
7560 IF G( 1 , 1 ) = 0 THEN 7660
7570 REM Ziehen
7580  V = I( V1 - 48 , V0 - 64 )
7590  N = I( N1 - 48 , N0 - 64 )
7600  E = 0
7610 REM
7620 IF V = 0 THEN 7970
7630 IF N = 0 THEN 7970
7640 REM
7650 GOTO 7830
7660 REM Setzen und Entfernen eines Steines
7670  V = 0
7680  N = I( V1 - 48 , V0 - 64 )
7690  E = I( N1 - 48 , N0 - 64 )
7700 REM
7710 IF N = 0 THEN 7970
7720 IF E = 0 THEN 7970
7730 REM
7740 GOTO 7830
7750 REM
7760 REM Setzen eines Steines
7770 REM
7780  V = 0
7790  N = I( V1 - 48 , V0 - 64 )
7800  E = 0
7810 REM
7820 IF N = 0 THEN 7970
7830 REM
7840 REM Test, ob der Zug erlaubt ist
7850 REM
7860  P = 1
7870 IF G( 1 , P ) <> V THEN 7940
7880 IF G( 2 , P ) <> N THEN 7940
7890 IF G( 3 , P ) <> E THEN 7940
7900 REM Der Zug ist erlaubt, ausfuehren.
7910 PRINT
7920 GOSUB 3500
7930 GOTO 6000
7940 REM Zug nicht erlaubt ?
7950  P = P + 1
7960 IF P <= G THEN 7870
7970 REM Der Zug ist verboten, neuen Zug abfragen.
7980 PRINT
7990 GOTO 7100
8000 REM Feldausgabe
8010 PRINT
8020 PRINT "    A  B  C  D  E  F  G"
8030 PRINT
8040  I = 0
8050 PRINT " 1  " ;
8060 GOSUB 8900
8070 PRINT "---------" ;
8080 GOSUB 8900
8090 PRINT "---------" ;
8100 GOSUB 8900
8110 PRINT "  1"
8120 PRINT "    |        |        |"
8130 PRINT " 2  |  " ;
8140 GOSUB 8900
8150 PRINT "-----" ;
8160 GOSUB 8900
8170 PRINT "-----" ;
8180 GOSUB 8900
8190 PRINT "  |  2"
8200 PRINT "    |  |     |     |  |"
8210 PRINT " 3  |  |  " ;
8220 GOSUB 8900
8230 PRINT "--" ;
8240 GOSUB 8900
8250 PRINT "--" ;
8260 GOSUB 8900
8270 PRINT "  |  |  3"
8280 PRINT "    |  |  |     |  |  |"
8290 PRINT " 4  " ;
8300 GOSUB 8900
8310 PRINT "--" ;
8320 GOSUB 8900
8330 PRINT "--" ;
8340 GOSUB 8900
8350 PRINT "     " ;
8360 GOSUB 8900
8370 PRINT "--" ;
8380 GOSUB 8900
8390 PRINT "--" ;
8400 GOSUB 8900
8410 PRINT "  4"
8420 PRINT "    |  |  |     |  |  |"
8430 PRINT " 5  |  |  " ;
8440 GOSUB 8900
8450 PRINT "--" ;
8460 GOSUB 8900
8470 PRINT "--" ;
8480 GOSUB 8900
8490 PRINT "  |  |  5"
8500 PRINT "    |  |     |     |  |"
8510 PRINT " 6  |  " ;
8520 GOSUB 8900
8530 PRINT "-----" ;
8540 GOSUB 8900
8550 PRINT "-----" ;
8560 GOSUB 8900
8570 PRINT "  |  6"
8580 PRINT "    |        |        |"
8590 PRINT " 7  " ;
8600 GOSUB 8900
8610 PRINT "---------" ;
8620 GOSUB 8900
8630 PRINT "---------" ;
8640 GOSUB 8900
8650 PRINT "  7"
8660 PRINT
8670 PRINT "    A  B  C  D  E  F  G"
8680 PRINT
8690 RETURN
8900 REM Ausgabe eines Steines
8910  I = I + 1
8920 ON F( I ) + 2 GOTO 8940 , 8960 , 8980
```

```
8940 PRINT "O" ;
8950 RETURN
8960 PRINT " " ;
8970 RETURN
8980 PRINT "@" ;
8990 RETURN
9000 REM
9010 REM Definition der Nachbarn
9020 REM
9030 DATA 2 ,  2 , 10 ,  0 ,  0
9040 DATA 3 ,  1 ,  3 ,  5 ,  0
9050 DATA 2 ,  2 , 15 ,  0 ,  0
9060 DATA 2 ,  5 , 11 ,  0 ,  0
9070 DATA 4 ,  2 ,  4 ,  6 ,  8
9080 DATA 2 ,  5 , 14 ,  0 ,  0
9090 DATA 2 ,  8 , 12 ,  0 ,  0
9100 DATA 3 ,  5 ,  7 ,  9 ,  0
9110 DATA 2 ,  8 , 13 ,  0 ,  0
9120 DATA 3 ,  1 , 11 , 22 ,  0
9130 DATA 4 ,  4 , 10 , 12 , 19
9140 DATA 3 ,  7 , 11 , 16 ,  0
9150 DATA 3 ,  9 , 14 , 18 ,  0
9160 DATA 4 ,  6 , 13 , 15 , 21
9170 DATA 3 ,  3 , 14 , 24 ,  0
9180 DATA 2 , 12 , 17 ,  0 ,  0
9190 DATA 3 , 16 , 18 , 20 ,  0
9200 DATA 2 , 13 , 17 ,  0 ,  0
9210 DATA 2 , 11 , 20 ,  0 ,  0
9220 DATA 4 , 17 , 19 , 21 , 23
9230 DATA 2 , 14 , 20 ,  0 ,  0
9240 DATA 2 , 10 , 23 ,  0 ,  0
9250 DATA 3 , 20 , 22 , 24 ,  0
9260 DATA 2 , 15 , 23 ,  0 ,  0
9270 REM
9280 REM Definition der Muehlen
9290 REM
9300 DATA 1 ,  1 ,  2 ,  3
9310 DATA 2 ,  4 ,  5 ,  6
9320 DATA 3 ,  7 ,  8 ,  9
9330 DATA 4 , 10 , 11 , 12
9340 DATA 5 , 13 , 14 , 15
9350 DATA 6 , 16 , 17 , 18
9360 DATA 7 , 19 , 20 , 21
9370 DATA 8 , 22 , 23 , 24
9380 DATA 9 ,  1 , 10 , 22
9390 DATA 0 ,  2 ,  5 ,  8
9400 DATA 1 ,  3 , 15 , 24
9410 DATA 2 ,  4 , 11 , 19
9420 DATA 3 ,  6 , 14 , 21
9430 DATA 4 ,  7 , 12 , 16
9440 DATA 5 ,  9 , 13 , 18
9450 DATA 6 , 17 , 20 , 23
9460 REM
9470 REM Extern -> Intern Wandlungstabelle
9480 REM
9490 DATA 0 ,  1 ,  0 ,  0 ,  2 ,  0 ,  0 ,  3
9500 DATA 0 ,  0 ,  4 ,  0 ,  5 ,  0 ,  6 ,  0
9510 DATA 0 ,  0 ,  0 ,  7 ,  8 ,  9 ,  0 ,  0
9520 DATA 0 , 10 , 11 , 12 ,  0 , 13 , 14 , 15
9530 DATA 0 ,  0 ,  0 , 16 , 17 , 18 ,  0 ,  0
9540 DATA 0 ,  0 , 19 ,  0 , 20 ,  0 , 21 ,  0
9550 DATA 0 , 22 ,  0 ,  0 , 23 ,  0 ,  0 , 24
9560 REM
9570 REM Bewertungsfeld : Steine
9580 REM
9590 DATA 9.9 , 9.9 , 9.9
9600 DATA 8.0 , 6.5 , 5.0
9610 DATA 3.3 , 2.0 , 0.9
9620 REM
9630 REM Bewertungsfeld : offene Muehlen
9640 REM
9650 DATA 3.30 , 3.10 , 2.90 , 2.70
9660 DATA 2.50 , 2.30 , 2.10 , 1.90
9670 DATA 1.70 , 1.50 , 1.30 , 1.10
9680 DATA 0.90 , 0.70 , 0.43 , 0.20
9690 REM
9700 REM Bewertungsfeld : geschlossene Muehlen
9710 REM
9720 DATA 1.10 , 1.05 , 1.00 , 0.95
9730 DATA 0.90 , 0.85 , 0.80 , 0.75
9740 DATA 0.70 , 0.65 , 0.60 , 0.55
9750 DATA 0.50 , 0.45 , 0.40 , 0.18
```


Neu!
Hayden Software auf Kassetten!
Ab Lager • Solange Vorrat reicht

Sargon II
○ TRS-80 **98,–**
○ APPLE II **98,–**

Mikrocomputer Aided Design aktive Filter
○ CBM (PET) **66,–**
○ TRS-80 **66,–**
○ APPLE II **66,–**

Songs
○ APPLE II **58,–**

Revive
○ APPLE II **96,–**

Engineering Maths
○ CBM (PET) **68,–**
○ TRS-80 **68,–**
○ APPLE II **68,–**

Preise einschl. MwSt. Bei allen Bestellungen unter 100 DM Porto- und Verpackungsanteil 5 DM.
Lieferung erfolgt nur gegen Nachnahme.

Bestellkupon

Ich/Wir bestellen gegen Nachnahme

_____

_____

_____

_____

Lieferanschrift: _____

_____

_____

_____
Datum Unterschrift

Bestellungen an: **CHIP-Software-Service**
z.Hd. Frau Barrois
Postfach 67 40
8700 Würzburg 1

Gerd S. Gottschling

Würfelspiel mit dem TI-59/PC 100C

Beschreibung

Das vorliegende Programm (Bild 3) berechnet durch Vorgabe eines Anfangswertes drei Zufallszahlen x, y, z ($1 \leq x, y, z \leq 6$) und interpretiert diese als „Wurf".

Der Spieler hat je Spiel maximal drei „Würfe", der Rechner maximal die vorgelegte Wurfanzahl des Spielers sofern erforderlich.

Der Spieler gewinnt den gesetzten Spielbetrag nur, wenn der erreichte Rang höher als der des Rechners ist. Für die verschiedenen Kombinationsmöglichkeiten des Wurfergebnisses wurde eine Rangfolge der militärischen Dienstgrade gewählt (Bild 1).

Beispiel 1

```
 WHERE'S YOUR MONEY
  9.9999999 99?

      O.K.

   GAME OF DICE
       1000.      CAP
--------------------

     PLAY ABOUT
        250.
 YOUR 1ST THROW
          1.3      LIE
                  2ND?
 YOUR 2ND THROW
         63.2     3RD?
 YOUR 3RD THROW
           2.      MAJ
                  LAST
      O.K.

   MY 1ST THROW
           6.      MAJ

         750.     LOST
--------------------
```

Beispiel 2

```
     PLAY ABOUT
        500.
 YOUR 1ST THROW
         63.1     2ND?
 YOUR 2ND THROW
         63.2     3RD?
 YOUR 3RD THROW
          2.3      LIE
                  LAST
     RIGHT

         250.     LOST
--------------------
```

Beispiel 3

```
     PLAY ABOUT
        250.
 YOUR 1ST THROW
         64.1     2ND?
 YOUR 2ND THROW
         53.2     3RD?
 YOUR 3RD THROW
          6.5      LIE
                  LAST
      O.K.

   MY 1ST THROW
          6.4      LIE
   MY 2ND THROW
         63.2
   MY 3RD THROW
          4.1      LIE

         500.      WON
--------------------
```

Beispiel 4

```
     PLAY ABOUT
        500.
 YOUR 1ST THROW
         43.2     2ND?
 YOUR 2ND THROW
         64.3     3RD?
      O.K.

   MY 1ST THROW
          1.3      LIE

           0.     LOST

    SORRY NO MONEY
 9.9999999 99?
```

HIERARCHY

1. GENERAL

```
DREI GLEICHE
ZAHLEN X, X, X
(1≤X≤6)

RANG:
      6.
      5.
      4.
      3.
      2.
      1.

   DRUCK  'X. GEN'
--------------------
```

2. MAJOR

```
ZAHLEN 1, 2, X
(1≤X≤6)

RANG:
      6.
      5.
      4.
      3.
      2.
      1.

   DRUCK  'X. MAJ'
```

3. LIEUTENANT

```
ZWEI GLEICHE
ZAHLEN    X, X
UND          Y
(1≤X, Y≤6)
  (X≠Y)

RANG:
6.5/6.4/6.3/
6.2/6.1/5.6/
5.4/5.3/5.2/
5.1/4.6/4.5/
4.3/4.2/4.1/
3.6/3.5/3.4/
3.2/3.1/2.6/
2.5/2.4/2.3/
1.6/1.5/1.4/
1.3/

   DRUCK 'X.Y LIE'
--------------------
```

4. (NO NAME)

```
VERSCHIEDENE
ZAHLEN X, Y, Z
(X>Y>Z)

RANG:
65.4/65.3/
65.2/65.1/
64.3/64.2/
64.1/63.2/
63.1/54.3/
54.2/54.1/
53.2/53.1/
43.2/43.1/

   DRUCK    'XY.Z'
```

Bild 1

Nr.	Verfahren	Eingabe	Taste	Anzeige	Bemerkung
0.	Programmeingabe bzw. Kartenlesung (639.39)				4 OP 17 (4 Blöcke) (Blöcke 3 u. 4 auch für Pgm. "place a no.")
1.	Programmstart				
1a.	Eingabe des Spieler - kapitals	.....DM	STO 11	.....DM	kein Spiel wenn die Eingabe nicht erfolgt ("WHERE'S .....")
1b.	Eingabe einer Zahl	0 < z < 1	A	0	Druck ("O.K. , GAME ..... ")
2.	Spiel				
2a.	Eingabe des Spielein - satzbetrages	.....DM	B	0	Druck ("PLAY ... ,YOUR ... THROW") "2ND?" wenn ja , Schritt 2b.
2b.	Wunsch 2. bzw. 3. Wurf		C	0	Druck ("YOUR ... THROW ") "3RD?" wenn ja , Schritt 2b. wenn nein oder "LAST" , 2c.
2c.	kein 2. bzw. 3. Wurf		D	0	Druck ("O.K. , MY ... THROW ") Berechnung , Vergleich und Druck ("... LOST / WON ") Wurfanzahl des Rechners maximal gleich der des Spielers , sofern erforderlich
2d.	Aufgabe nach dem letzten Wurf		E	0	Druck ("RIGHT ... LOST")
3.	Wiederholung 2.				neues Spiel (solange R11>0)

Bild 2

Interpretation des Vorgangs (Bild 2) und der Beispiele

Nach Durchführung der Schritte 0. / 1a. / 1b. / weist die Null in der Anzeige auf die Spielbereitschaft hin. Auf dem Druckstreifen wird die richtige Handhabung bestätigt („O.K."), die Überschrift und das Spielkapital gedruckt (im Beispiel 1 zu Beginn Schritt 1a. vergessen, „WHERE'S YOUR MONEY 999 ... ?").

Das Spiel beginnt mit der Eingabe des Spieleinsatzbetrages, Schritt 2a. und dem Start mit [B].

Der erste „Wurf" des Spielers wird berechnet, der Spielbetrag, Wurfhinweis und das Wurfergebnis gedruckt und mit „2ND?" gefragt, ob ein zweiter „Wurf" gewünscht wird.

In den Spielen wird positiv durch [C], negativ durch [D] geantwortet.

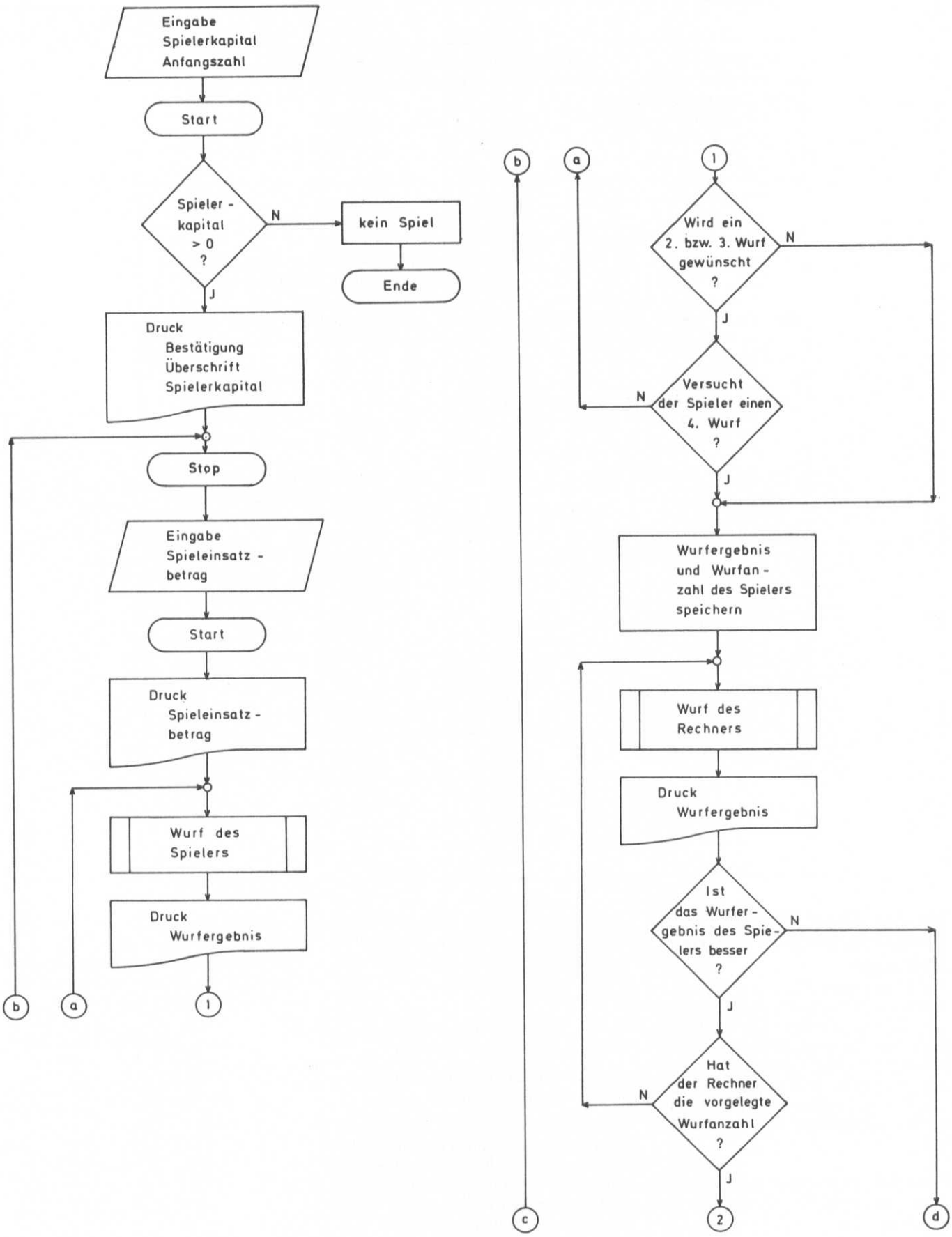
Mit [D] wird dann die Berechnung der Rechnerwürfe eingeleitet.

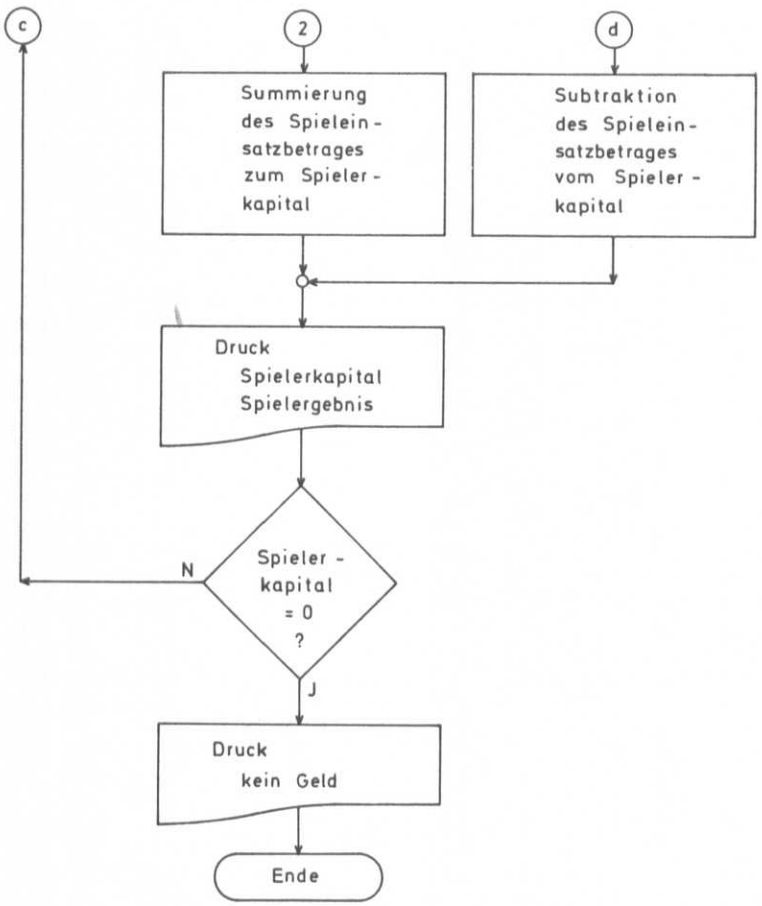
Im Beispiel 1 erreichte der Rechner schon mit dem ersten Wurf einen höheren Rang als der Spieler, im Beispiel 2 hat der Spieler mit [E] nach dem dritten und letzten Wurf aufgegeben (keine Berechnung für den Rechner).

Im Beispiel 3 erreichte der Rechner auch mit dem dritten Wurf nicht den vom Spieler vorgelegten Rang, im Beispiel 4 ist das Spielerkapital verspielt (= 0) und das Spiel beendet.

(bitte umblättern)

Würfelspiel mit dem TI-59/PC 100C





UP Wurf
1. Berechnung von drei Zufallszahlen
2. Sortierung
3. Rangbestimmung
4. Zuordnung

```
000  76 LBL
001  69 OP
002  69 OP
003  05  05
004  69 OP
005  00  00
006  25 CLR
007  92 RTN
008  76 LBL
009  11  A
010  42 STO
011  05  05
012  29 CP
013  22 INV
014  86 STF
015  00  00
016  43 RCL
017  11  11
018  67  EQ
019  45 Y^x
020  43 RCL
021  31  31
022  69 OP
023  02  02
024  71 SBR
025  69 OP
026  98 ADV
027  43 RCL
028  12  12
029  69 OP
030  01  01
031  43 RCL
032  13  13
033  69 OP
034  02  02
035  43 RCL
036  14  14
037  69 OP
038  03  03
039  71 SBR
040  69 OP
041  43 RCL
042  15  15
043  69 OP
044  04  04
045  43 RCL
046  11  11
047  69 OP
048  06  06
049  43 RCL
050  16  16
051  69 OP
052  01  01
053  69 OP
054  02  02
055  69 OP
056  03  03
057  69 OP
058  04  04
059  71 SBR
060  69 OP
061  98 ADV
062  02  2
063  01  1
064  42 STO
065  06  06
066  71 SBR
067  60 DEG
068  71 SBR
069  70 RAD
070  25 CLR
071  91 R/S
072  76 LBL
073  12  B
074  42 STO
075  10  10
076  43 RCL
077  23  23
078  69 OP
079  02  02
080  43 RCL
081  24  24
082  69 OP
083  03  03
084  71 SBR
085  69 OP
086  43 RCL
087  10  10
088  99 PRT
089  69 OP
090  00  00
091  73 RC*
092  06  06
093  69 OP
094  01  01
095  73 RC*
096  07  07
097  69 OP
098  02  02
099  43 RCL
100  25  25
101  69 OP
102  03  03
103  71 SBR
104  69 OP
105  29 CP
106  69 OP
107  27  27
108  43 RCL
109  04  04
110  55  ÷
111  01  1
112  01  1
113  01  1
114  95  =
115  42 STO
116  00  00
117  22 INV
118  59 INT
119  67  EQ
120  38 SIN
121  71 SBR
122  88 DMS
123  43 RCL
124  01  01
125  65  ×
126  01  1
127  00  0
128  00  0
129  95  =
130  42 STO
131  04  04
132  43 RCL
133  02  02
134  65  ×
135  01  1
```

Bild 3

(bitte umblättern)

Würfelspiel mit dem TI-59/PC 100C

```
136  00  0
137  95  =
138  44  SUM
139  04  04
140  43  RCL
141  03  03
142  44  SUM
143  04  04
144  02  2
145  01  1
146  32  X↕T
147  71  SBR
148  80  GRD
149  67  EQ
150  42  STO
151  71  SBR
152  90  LST
153  67  EQ
154  39  COS
155  29  CP
156  71  SBR
157  90  LST
158  55  ÷
159  01  1
160  01  1
161  95  =
162  42  STO
163  02  02
164  22  INV
165  59  INT
166  67  EQ
167  43  RCL
168  71  SBR
169  80  GRD
170  55  ÷
171  01  1
172  01  1
173  95  =
174  42  STO
175  01  01
176  22  INV
177  59  INT
178  67  EQ
179  44  SUM
180  69  OP
181  00  00
182  87  IFF
183  00  00
184  01  01
185  94  94
186  73  RC*
187  07  07
188  85  +
189  07  7
190  01  1
191  95  =
192  69  OP
193  04  04
194  43  RCL
195  04  04
196  55  ÷
197  01  1
198  00  0
199  95  =
200  42  STO
201  01  01
202  69  OP
203  06  06
204  25  CLR
205  69  OP
206  00  00
207  87  IFF
208  00  00
209  02  02
210  54  54
211  91  R/S
212  76  LBL
213  13  C
214  00  0
215  42  STO
216  04  04
217  03  3
218  42  STO
219  00  00
220  02  2
221  00  0
222  32  X↕T
223  43  RCL
224  07  07
225  77  GE
226  14  D
227  71  SBR
228  70  RAD
229  61  GTO
230  00  00
231  89  89
232  76  LBL
233  14  D
234  43  RCL
235  31  31
236  69  OP
237  02  02
238  71  SBR
239  69  OP
240  86  STF
241  00  00
242  43  RCL
243  01  01
244  42  STO
245  08  08
246  43  RCL
247  07  07
248  42  STO
249  09  09
250  71  SBR
251  60  DEG
252  61  GTO
253  70  RAD
254  43  RCL
255  08  08
256  32  X↕T
257  43  RCL
258  01  01
259  77  GE
260  23  LNX
261  43  RCL
262  09  09
263  32  X↕T
264  43  RCL
265  07  07
266  77  GE
267  24  CE
268  69  OP
269  36  36
270  03  3
271  42  STO
272  00  00
273  25  CLR
274  42  STO
275  04  04
276  61  GTO
277  70  RAD
278  76  LBL
279  22  INV
280  73  RC*
281  07  07
282  85  +
283  07  7
284  01  1
285  95  =
286  69  OP
287  04  04
288  71  SBR
289  69  OP
290  25  CLR
291  91  R/S
292  76  LBL
293  23  LNX
294  98  ADV
295  43  RCL
296  10  10
297  22  INV
298  44  SUM
299  11  11
300  43  RCL
301  29  29
302  69  OP
303  04  04
304  61  GTO
305  35  1/X
306  76  LBL
307  24  CE
308  98  ADV
309  43  RCL
310  10  10
311  44  SUM
312  11  11
313  43  RCL
314  30  30
315  69  OP
316  04  04
317  76  LBL
318  35  1/X
319  43  RCL
320  11  11
321  69  OP
322  06  06
323  22  INV
324  86  STF
325  00  00
326  29  CP
327  43  RCL
328  11  11
329  68  NOP
330  67  EQ
331  78  Σ+
332  61  GTO
333  00  00
334  49  49
335  76  LBL
336  42  STO
337  43  RCL
338  02  02
339  22  INV
340  59  INT
341  65  ×
342  01  1
343  00  0
344  95  =
345  42  STO
346  01  01
347  61  GTO
348  39  COS
349  76  LBL
350  43  RCL
351  43  RCL
352  01  01
353  59  INT
354  55  ÷
355  01  1
356  00  0
357  95  =
358  42  STO
359  01  01
360  43  RCL
361  02  02
362  44  SUM
363  01  01
364  61  GTO
365  30  TAN
366  76  LBL
367  44  SUM
368  43  RCL
369  02  02
370  22  INV
371  59  INT
372  44  SUM
373  01  01
374  61  GTO
375  30  TAN
376  76  LBL
377  38  SIN
378  43  RCL
379  26  26
380  69  OP
381  04  04
382  43  RCL
383  00  00
384  69  OP
385  06  06
386  65  ×
387  04  4
388  22  INV
389  28  LOG
390  95  =
391  42  STO
392  01  01
393  87  IFF
394  00  00
395  02  02
396  54  54
397  61  GTO
398  22  INV
399  76  LBL
400  39  COS
401  43  RCL
402  27  27
403  69  OP
404  04  04
405  43  RCL
406  01  01
407  59  INT
408  69  OP
409  06  06
410  65  ×
411  03  3
412  22  INV
413  28  LOG
414  95  =
415  42  STO
416  01  01
417  87  IFF
418  00  00
419  02  02
420  54  54
421  61  GTO
422  22  INV
423  76  LBL
424  30  TAN
425  43  RCL
426  28  28
427  69  OP
428  04  04
429  43  RCL
430  01  01
431  69  OP
432  06  06
433  65  ×
434  01  1
435  00  0
436  00  0
437  95  =
438  42  STO
439  01  01
440  87  IFF
441  00  00
442  02  02
443  54  54
444  61  GTO
445  22  INV
446  76  LBL
447  60  DEG
448  03  3
449  42  STO
450  00  00
451  01  1
452  07  7
453  42  STO
454  07  07
455  00  0
456  42  STO
457  04  04
458  92  RTN
459  76  LBL
460  70  RAD
461  71  SBR
462  89  π
463  72  ST*
464  00  00
465  65  ×
466  43  RCL
467  00  00
468  22  INV
469  28  LOG
470  95  =
471  44  SUM
472  04  04
473  97  DSZ
474  00  00
475  70  RAD
476  01  1
477  00  0
478  22  INV
479  49  PRD
480  04  04
```

```
481  87 IFF
482  00  00
483  79  x̄
484  92 RTN
485  76 LBL
486  80 GRD
487  43 RCL
488  04  04
489  55  ÷
490  01  1
491  00  0
492  95  =
493  42 STO
494  02  02
495  59 INT
496  92 RTN
497  76 LBL
498  90 LST
499  43 RCL
500  04  04
501  55  ÷
502  01  1
503  00  0
504  00  0
505  95  =
506  42 STO
507  01  01
508  22 INV
509  59 INT
510  65  ×
511  01  1
512  00  0
513  00  0
514  95  =
515  92 RTN
516  76 LBL
517  78 Σ+
518  98 ADV
519  69 OP
520  00  00
521  43 RCL
522  32  32
523  69 OP
524  02  02
525  43 RCL
526  33  33
527  69 OP
528  03  03
529  43 RCL
530  34  34
531  69 OP
532  04  04
533  71 SBR
534  69 OP
535  35 1/X
536  99 PRT
537  98 ADV
538  91 R/S
539  76 LBL
540  79  x̄
541  69 OP
542  26  26
543  61 GTO
544  00  00
545  89  89
546  76 LBL
547  88 DMS
548  00  0
549  42 STO
550  04  04
551  43 RCL
552  02  02
553  32 X↕T
554  43 RCL
555  01  01
556  22 INV
557  77  GE
558  57 ENG
559  43 RCL
560  03  03
561  32 X↕T
562  43 RCL
563  02  02
564  22 INV
565  77  GE
566  58 FIX
567  92 RTN
568  76 LBL
569  89  π
570  89  π
571  85  +
572  43 RCL
573  05  05
574  95  =
575  33  X²
576  22 INV
577  59 INT
578  65  ×
579  06  6
580  95  =
581  42 STO
582  05  05
583  59 INT
584  85  +
585  01  1
586  95  =
587  92 RTN
588  76 LBL
589  45 Y^x
590  43 RCL
591  35  35
592  69 OP
593  01  01
594  43 RCL
595  36  36
596  69 OP
597  02  02
598  43 RCL
599  21  21
600  65  ×
601  01  1
602  00  0
603  00  0
604  95  =
605  69 OP
606  03  03
607  61 GTO
608  05  05
609  29  29
610  76 LBL
611  57 ENG
612  43 RCL
613  02  02
614  48 EXC
615  01  01
616  48 EXC
617  02  02
618  61 GTO
619  88 DMS
620  76 LBL
621  58 FIX
622  43 RCL
623  03  03
624  48 EXC
625  02  02
626  48 EXC
627  03  03
628  61 GTO
629  88 DMS
630  76 LBL
631  15  E
632  43 RCL
633  37  37
634  69 OP
635  02  02
636  71 SBR
637  69 OP
638  61 GTO
639  23 LNX
```

REGISTER		LABELS		
INTERN	0÷9	001	69	OP
EINSATZ	10	009	11	A
KAPITAL	11	073	12	B
2213.	12	213	13	C
3017003221.	13	233	14	D
16241517.	14	279	22	INV
151333.	15	293	23	LNX
2020202020.	16	307	24	CE
2363700.	17	318	35	1/X
3311600.	18	336	42	STO
4351600.	19	350	43	RCL
27133566.	20	367	44	SUM
45324135.	21	377	38	SIN
3045.	22	400	39	COS
3327134500.	23	424	30	TAN
1314324137.	24	447	60	DEG
3723353243.	25	460	70	RAD
221731.	26	486	80	GRD
301325.	27	498	90	LST
272417.	28	517	78	Σ+
27323637.	29	540	79	x̄
433231.	30	547	88	DMS
32402640.	31	569	89	π
36323535.	32	589	45	Y^x
4500313200.	33	611	57	ENG
3032311745.	34	621	58	FIX
432317.	35	631	15	E
3517653600.	36			
3524222337.	37			
0.	38			
0.	39			

Bernhard Engl

ELIZA II für 6502-Systeme

Beschreibung

Dieses Programm ist ein mehr oder weniger typischer Vertreter aus dem Gebiet der künstlichen Intelligenz (engl.: „Artifical Intelligence"). Diese Wissenschaft wird von einem ihrer Schöpfer, Marvin Minsky vom Massachusetts Institute of Technology (MIT) wie folgt definiert: „Artificial Intelligence ist die Wissenschaft, Maschinen Dinge tun zu lassen, die Intelligenz erforderten, wenn Menschen sie tun müßten." Das bekannteste Beispiel hierfür sind seit Jahren die Schachcomputer, gegen die sogar Großmeister schon Schwierigkeiten hatten. Jedoch ist es nicht die Absicht der AI-Forschung, alle nur irgendwie erreichbaren Intelligenz- und Taktikspiele zu „computerifizieren". Das große Ziel der AI-Forschung ist es, ein Computerprogramm zu schaffen, welches in der Lage ist, unsere Alltagssprache zu verstehen und gegebenenfalls auch in dieser Sprache zu antworten. Solange wir uns mit Computern in „ihrer" Sprache unterhalten müssen, solange werden unsere Möglichkeiten, mit ihnen in Verbindung zu treten, begrenzt sein. Warum also lehren wir sie nicht einfach unsere Alltagssprache? Nun, wer sich schon einmal etwas damit befaßt hat, wird zugeben müssen, daß dies komplizierter ist, als es scheint. Einen der ersten Schritte unternahm Mitte der 60er Jahre Joseph Weizenbaum vom MIT. Er brachte einem Computer bei, sich durch eine Unterhaltung zu mogeln, indem er Rückfragen stellte und immer wieder um Erklärungen und genauere Angaben bat. Dies ist aber nur dann möglich, wenn man sich auf ein bestimmtes Gebiet beschränkt, der Wortschatz des Computers muß darauf zugeschnitten sein. Weizenbaum wählte daher eine einfachere Rolle für den Computer aus, auch, um das Vokabular in Grenzen zu halten. Sein in der Programmiersprache LISP geschriebenes Programm spielt einen „nichtdirektiven" Psychotherapeuten, der einen verstörten Patienten befragt.

Dieser Artikel soll dem Leser die Möglichkeit geben, eine einfache Version dieses Programms auf seinem Mikrocomputer zu realisieren, ohne eine höhere Programmiersprache benutzen zu müssen, da diese zuviel Speicherplatz benötigen würde.

Sich mit einem Computer zu unterhalten als wäre er ein menschliches Wesen, gehört sicherlich zu den reizvollsten und eindrucksvollsten Vorführobjekten eines Hobbyisten. ELIZA II, so heißt das vorliegende Programm, stellt einen Psychoanalytiker dar, dem man seine Probleme anvertrauen kann, allerdings nur auf Englisch, da sich diese Sprache durch einen einfachen Aufbau auszeichnet. Es gibt die verschiedensten Variationsmöglichkeiten bezüglich des Themas, erreichbar durch eine einfache Veränderung einer Texttabelle. So kann z.B. ohne einen Eingriff in das Hauptprogramm erreicht werden, daß der Computer über den Sinn oder Unsinn des Computerhobbys diskutieren kann, ... der Phantasie sind keine Grenzen gesetzt.

Es gibt bereits mehrere ELIZA-Programme von verschiedenen Autoren, das Vorliegende ist bisher das Schnellste und vom Speicherplatz her das Anspruchloseste. Dieses Programm wurde auf einem Superboard mit 4K-RAM entwickelt, läßt sich aber auch an andere 6502-Systeme anpassen. Hat man mehr Speicher zur Verfügung, so ist eine Erweiterung des Wortschatzes ohne weiteres möglich.

Wie man aus dem Flußdiagramm (Bild 1) entnehmen kann, beruht die Funktion auf einer Schlüsselwortsuche. Diese Schlüsselwortsuche ermöglicht es dem Programm, zu einem beliebigen, eingegebenen Satz einen passenden Antwortsatz zu finden. Steht am Ende des Antwortsatzes ein Sternchen, so wird der Eingabesatz bis zum Ende des Schlüsselwortes gelöscht, dann konjugiert (so daß aus den Worten „I am" die Wörter „you are" werden usw.) und an den Antwortsatz angefügt.

Die Qualität der ausgegebenen Sätze steht und fällt mit der Qualität der Texttabelle (Bild 3), diese ist daher sorgfältig einzugeben. Leerzeichen, welche unbedingt mit einzugeben sind, sind unterstrichen (Syntax siehe Bild 5). Eventuelle Erweiterungssätze sind vor dem unterstrichenen Schlüsselwort einzufügen. Die Hilfsprogramme nach Bild 4 erleichtern die Eingabe des Programms (Bild 2) und der Texttabelle sowie die Aufzeichnung auf Band. (Zuerst das Programm, dann die Texttabelle, beide Aufzeichnungen rund 2 bis 3 s getrennt.)

Diese Hilfsprogramme sind neu zu schreiben, falls ELIZA auf einem anderen 6502-Computer laufen soll. Alle wichtigen Informationen sind im Bild 6 zusammengefaßt, alle höherwertigen Adressen sind in Bild 2 unterstrichen. Bild 7 zeigt eine „Sitzung" mit ELIZA, sie kann zu Testzwecken herangezogen werden.

Hat man sich längere Zeit mit „ELIZA II" unterhalten, so wird man feststellen, daß sich der Computer manchmal wiederholt. Der Wortschatz des Programms wurde so gewählt, daß er bei minimalem Speicherbedarf den größtmöglichen Effekt erzielt. (Man sollte nicht vergessen, das Programm dient nur zu Vorführzwecken.) Hat man mehr Speicher zur Verfügung, so kann man den Wortschatz leicht ausbauen, indem man für ein Schlüsselwort mehrere mögliche Antwortsätze hinzufügt. Dies ist sehr leicht, man muß nur nach dem bisher letzten Antwortsatz eines Schlüsselwortes die Folge: f ANTWORT eingeben, sonst ist nichts zu tun.

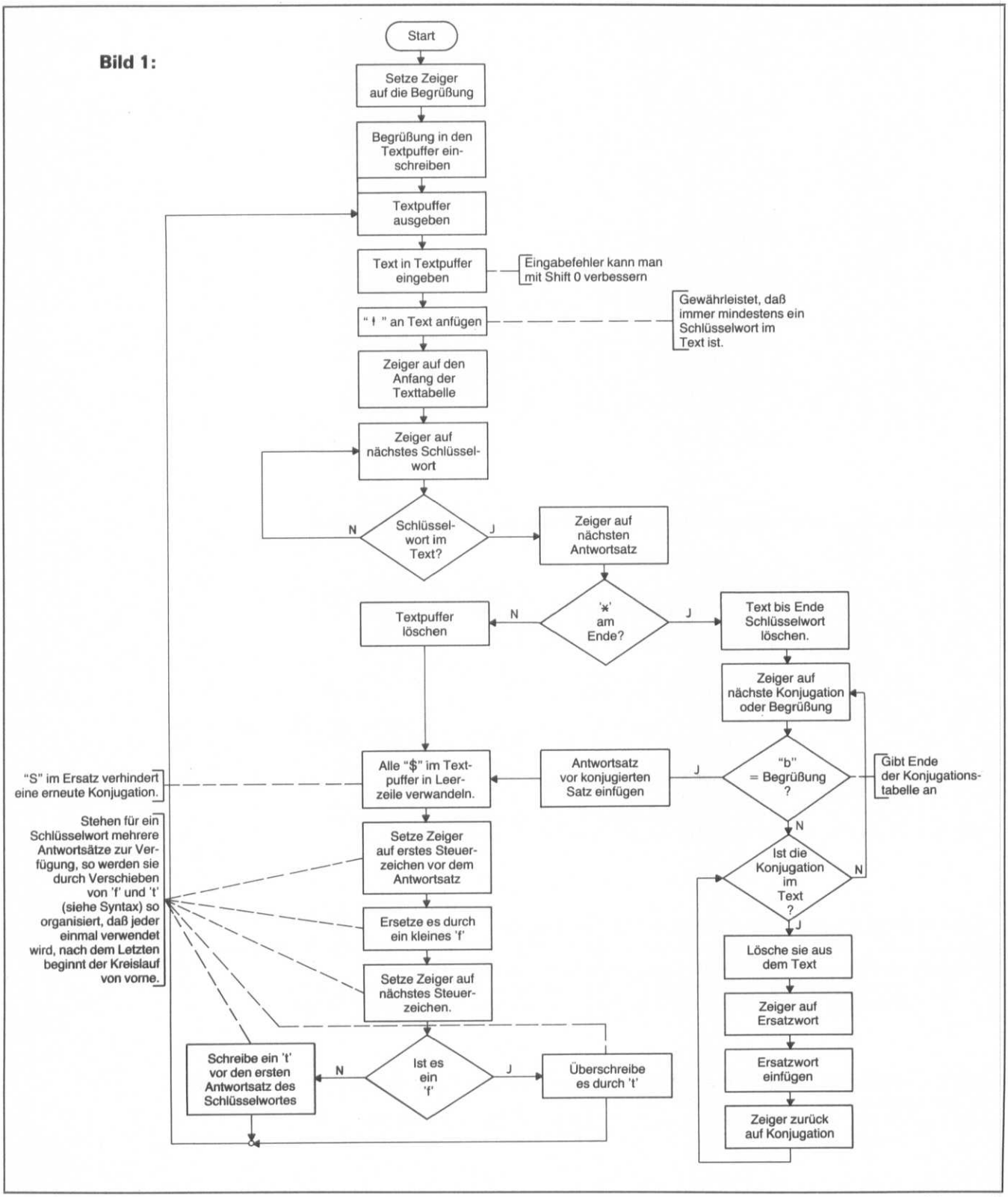
Es ist auch möglich, völlig neue Schlüsselwörter aufzunehmen. Diese fügt man am besten unmittelbar vor dem Null-Schlüsselwort (↑) ein. (Das Nullschlüsselwort markiert die Sätze, welche ausgegeben werden, falls kein anderes im Eingabesatz aufzufinden ist.) Als Anregung sei hier ein Beispiel genannt: die Schlüsselwörter „mother", „father", „sister", „brother" könnten als Antwort „do you have problems with your family" oder „tell me more about your family" zur Folge haben. (Da die Kleinbuchstaben als Steuerzeichen verwendet werden, sind die Sätze natürlich mit Großbuchstaben einzugeben.)

Wem die Unterhaltung auch dann noch zu „seicht" ist, sollte bedenken, daß das Programm keinen echten Psychoanaly-

tiker ersetzen kann und soll. Im Gegenteil, hier treten die Gefahren der AI offen zutage. Weizenbaum kämpft heute gegen den möglichen Mißbrauch der AI. Er bekam Bedenken, als Leute anfingen, den Computer ernst zu nehmen; manche verlangten sogar, mit dem Terminal allein zu sein, um ihm ihr Herz auszuschütten. Diese Tatsachen ändern aber nichts an dem Wert des Programms als eindrucksvolles Vorführobjekt.

Literatur:
[1] Creative Computing Juli/ August 77, S. 100 ff
[2] Funkschau Mikrocomputer Sonderheft 1, S. 95 ff

Bild 1:



(bitte umblättern)

ELIZA II für 6502-Systeme

Bild 2: Das Programm

```
0300 E6 54 D0 02 E6 55 60 48 A9 FF 18 65 54 85 54 B0 02 C6 55 68 60
     A0 00 20 00 03 B1 54 C9 60 30 F7 60 A0 00 20 07 03 B1 54 C9 60
     30 F7 60 A0 22 84 50 A2 00 86 52 A1 54 C9 60 10 21 D9 00 02 F0
     10 20 21 03 20 00 03 E6 50 A4 50 C4 51 D0 E3 18 60 E6 52 C8 C4
     51 F0 F7 20 00 03 38 B0 D8 38 60 A9 0D 20 2D BF A9 0A 20 2D BF
     A9 0A 20 2D BF 60 20 15 03 20 07 03 A2 00 A1 54 C9 60 30 01 60
     C9 2A D0 02 A9 20 48 A4 51 88 B9 00 02 C8 99 00 02 88 C4 50 D0
     F3 E6 51 68 A4 50 99 00 02 38 B0 D3 A4 50 C8 C4 51 F0 0B B9 00
     02 88 99 00 02 C8 38 B0 F0 C6 51 C6 59 D0 E8 60 A2 22 86 50 BD
     00 02 20 2D BF E6 50 A6 50 E4 51 D0 F2 20 5F 03 60 A2 23 86 51
     20 ED FE 48 20 2D BF 68 C9 5F F0 0D C9 0D F0 0D A6 51 9D 00 02
     E6 51 D0 E7 C6 51 D0 FA 20 5F 03 60 A5 54 85 56 A5 55 85 57 60
     A5 56 85 54 A5 57 85 55 60 A9 DE 18 65 50 18 65 52 85 59 A9 22
     85 50 20 E5 04 60 A4 51 A9 5E 99 01 02 99 02 02 A9 20 99 00 02
     E6 51 E6 51 E6 51 A0 22 99 00 02 60 A0 22 B9 00 02 C9 24 D0 02
     A9 20 C9 5E D0 02 A9 20 99 00 02 C8 C4 51 D0 E9 60 20 38 05 20
     15 03 C9 6B D0 F9 20 00 03 20 2D 03 90 F1 20 07 03 20 15 03 C9
     74 D0 F9 20 00 03 60 20 07 03 20 15 03 C9 65 F0 09 C9 67 D0 F5
     20 00 03 38 60 18 60 20 2D 03 B0 01 60 A5 52 85 59 20 9F 03 20
     00 03 20 6F 03 20 21 03 20 21 03 20 00 03 38 B0 E1 20 F3 03 20
     07 03 A9 66 A0 00 91 54 20 15 03 C9 66 F0 0A 20 21 03 C9 6B D0
     F9 20 15 03 A9 74 91 54 60 A9 00 85 54 A9 07 85 55 20 21 03 C9
     78 D0 F9 20 15 03 C9 62 D0 F9 20 00 03 20 F3 03 A9 22 85 50 85
     51 60 20 F3 03 20 40 05 B1 54 C9 2A F0 07 A9 DB 18 65 51 85 59
     20 FC 03 20 9F 03 60 20 5F 03 20 C2 04 20 FC 03 20 6F 03 20 32
     04 20 B8 03 20 CE 03 20 17 04 20 4C 04 20 05 04 20 A0 04 A5 51
     C9 22 F0 0E C9 23 F0 0B 20 6C 04 90 06 20 81 04 38 B0 F5 4C 47
     05 20 21 03 C9 78 D0 F9 60 20 15 03 20 07 03 60 A9 22 85 50 4C
     05 05
```

Bild 4a: BASIC-Hilfsprogramm zum Aufzeichnen des ELIZA II-Programms

```
 10 DIM A(589)
 20 REM EINGABE DES HAUPTPROGRAMMS
 30 FOR I=0 TO 589
 40 INPUT A$:IF A$="XX"THEN I=I-1:GOTO40:REM FEHLERVERBESSERUNG
 50 IF LEN(A$) <>2THEN ?"WAS SOLL DAS?":GOTO40
 60 H= ASC(LEFT$(A$,1))-48:IFH>9THENH=H-7
 70 L=ASC(RIGHT$(A$,1))-48:IFL>9THENL=L-7
 80 A(I)=H*16+L:NEXT
 90 ?"RECORDER STARTEN, DANN GOTO 100":STOP
100 SAVE:?".000F/48":REM ZEILENBREITE
110 ?".0200/65":REM CURSOR
120 FOR I=1TO26:READ A$:?A$:NEXT:REM LADEPROGRAMM FUER TEXTTABELLE
130 B$="0123456789ABCDEF":?".0300/E6":FOR I=1TO589:X=A(I)
140 A$=MID$(B$,INT(X/16)+1,1)+MID$(B$,X-INT(X/16)*16+1,1)
150 ?A$:REM HAUPTPROGRAMM AUF BAND
160 NEXT:?".0222G":LOAD:END
170 DATA .005B/4E,05,.0222/20,80,FE,A0,00,91,5B,E6,5B,D0,02
180 DATA E6,5C,C9,65,D0,EF,A9,00,85,FB,4C,FF,04
```

Bild 4b: Hilfsprogramm zur Eingabe der Texttabelle

Startadresse = 0222, Anhalter nur durch „BREAK" möglich, SHIFT 0 wirksam. Vorher einen einmaligen BASIC-Kaltstart ausführen, da die BASIC-Ausgaberoutine benutzt wird!

```
0222 A9 4E 85 5B A9 05 85 5C 20 ED FE 48 20 2D BF 68 C9 5F D0 0E A9 FF 18 65
     5B 85 5B B0 EB C6 5C 4C 2A 02 A0 00 91 5B E6 5B D0 02 E6 5C 4C 2A 02
```

Bild 4c: Hilfsprogramm zum Aufzeichnen der Texttabelle auf Band,

Startadresse = 0222, Anhalten durch „BREAK"

```
0222 A9 4E 85 5B A9 05 85 5C 20 A6 FC A0 00 B1 5B 20 B1 FC B1 5B 20 2D BF
     B1 5B C9 65 F0 FE E6 5B D0 02 E6 5C 4C 2D 02
```

Bild 3: Die Texttabelle

```
xkCAN_YOU_tDON'T YOU BELIEVE THAT I CAN*fDO YOU THINK I
CAN'T*fYOU WANT ME TO BE ABLE TO*kCAN_I_tPERHAPS YOU
DON'T WANT TO*fDO YOU WANT TO BE ABLE TO*kYOU_ARE_kY-
OU'RE_tWATH MAKES YOU THINK I AM*fDOES IT PLEASE YOU TO
BELIEVE I AM*fPERHAPS YOU WOULD LIKE TO BE*kI__
<ON'TtDON'T YOU REALLY*fWHY DON'T YOU*fDO YOU WISH TO BE
ABLE TO*fDOES THAT TROUBLE YOU?kI_FEEL_tTELL ME MORE AB-
OUT SUCH FEELINGS.fDO YOU OFTEN FEEL*fDO YOU ENJOY FEE-
LING*kWHY_DON'T_YOU_kMAY_YOU_tDO YOU REALLY BELIEVE I
DON'T*fPERHAPS IN GOOD TIME I WILL*fDO YOU WANT ME TO*kW-
HY_CAN'T_I_tDO YOU THINK YOU SHOULD BE ABLE TO*fWHY
CAN'T YOU*kARE_YOU_tWHY ARE YOU INTERESTED WHETHER OR
NOT I AM*fWOULD YOU PREFER IT I WERE NOT*fPERHAPS IN YOUR
FANTASIES I AM*kI_CAN'TtHOW DO YOU KNOW YOU CAN'T*fHAVE
YOU TRIED?fPERHAPS YOU CAN NOW*kI_AM_kI'MtDID YOU COME
TO ME BECAUSE YOU ARE*fHOW LONG HAVE YOU BEEN*fDO YOU BE-
LIEVE IT IS NORMAL TO BE*kYOU_tOH,I*fYOU ARE NOT REALLY
TALKING ABOUT ME, ARE YOU?kI_WANT_tWHAT WOULD IT MEAN TO
YOU IF YOU GOT*fWHY DO YOU WANT*fSUPPOSE YOU SOON GOT*fI
SOMETIMES ALSO WANT*kWHATkHOWkWHOkWEREkWHENkWHY_tWHY
DO YOU ASK?fWHAT ANSWER WOULD YOU PLEASE THE MOST?fWHAT DO
YOU THINK?fARE SUCH QUESTIONS IN YOUR MIND OFTEN?fHAVE
YOU ASKED ANYONE ELSE?kNAMEtNAMES DON'T INTEREST
ME.kCAUSE_tIS THAT THE REAL REASON?fDON'T ANY OTHER REA-
SONS COME TO MIND?fARE YOU SURE?kSORRYtAPOLOGIES ARE NOT
NECESSARY.fWHAT FEELINGS DO YOU HAVE WHEN YOU APOLOGI-
ZE?fDON'T BE SO DEFENSIVE!kDREAMtDO YOU DREAM OF-
TEN?fWHAT PERSONS APPEAR IN YOUR DREAMS?kHI_kHELLOtHOW
DO YOU DO . . . PLEASE STATE YOUR PROBLEM!kMAYBEtYOU DON'T
SEEM QUITE CERTAIN!fWHY THE UNCERTAIN T ONE?fYOU AREN'T
SURE?k_NO_tWHY NO*fWHY NOT?fARE YOU SURE?fWHY NO?kY-
OURtWHY ARE YOU CONCERNED ABOUT MY*fWHAT ABOUT MY*kAL-
WAYStWHEN?fWHAT ARE YOU THINKING OF?REALLY, ALWAYS?kI__
THINKkI_FINDtWHAT MAKES YOU THINK*fARE YOU SURE?fDO YOU
DOUBT*kALIKEtIN WHAT WAY?fHOW?kYES_tYOU SEEM QUITE
POSITIVE.fARE YOU SURE?fI SEE.fI UNDERSTAND.kCOMPU-
TERtDO COMPUTERS WORRY YOU?fARE YOU TALKING ABOUT ME IN
PARTICULAR?kFRIENDtDO YOUR FRIENDS WORRY YOU?fDO YOUR
FRIEND PICK ON YOU? fARE YOU SURE YOU HAVE ANY FRIENDS?kMO-
NEYtDO YOU THINK MONEY IS EVERYTHING?fHAVE YOU PROBLEMS
WITH MONEY?ktWHAT DOES THAT SUGGEST TO YOU?fCAN YOU ELA-
BORATE ON THAT?fI SEE.fI UNDERSTAND THAT.fTHAT IS QUITE
INTERESTING.g_I_AM_t$YOU$ ARE_g_YOU_A-
RE_t$I$AM_g_ME_t$YOU_g_WERE_t$WAS_g_WAS_t$WE-
RE_g_I_t$YOU_g_YOUR_t$          MY_g_MY_t$YOUR_g-
WI'VE_t$YOU'VE_g_YOU'VE_t$I'VE_g_I'M_t$YOU'RE_g-
WYOU'RE_t$I'M_g_Y-
OU_t$ME_b_____________________________HI,
I'M ELIZA II TELL ME YOUR PROBLEM !e
```

Bedienung

Das Hauptproblem liegt darin, das Programm und die Texttabelle einzugeben und auf Band zu speichern.

Zunächst sei die Eingabe des Programms besprochen. Da der Monitor des Superboards über keine SAVE-Eingabe verfügt, muß dies mit einem BASIC-Hilfsprogramm geschehen (siehe Bild 4a). Dieses wird gestartet, dann wird das erste Byte des Programms eingegeben, dann das nächste usw. bis zum letzten. Hier sollte dann „RECORDER STARTEN . . ." auf dem Bildschirm erscheinen, sonst hat man ein Byte übersehen und muß von vorne anfangen. Eingabefehler kann man mit SHIFT 0 verbessern, solange noch nicht RETURN gedrückt wurde, ansonsten nur durch Eingabe von „XX", hierdurch wird das letzte Byte gelöscht und kann neu eingegeben werden. Hat man die Anweisung „RECORDER STARTEN . . ." erhalten und ausgeführt, tippt man „GOTO 100", und das Programm wird aufgezeichnet. (Man könnte auch „CONT" verwenden, was bei fehlerhafter Eingabe kein zweitesmal mehr funktioniert.) Das Hilfsprogramm zeichnet auch ein kleines Ladeprogramm mit auf, welches später automatisch gestartet wird und die Texttabelle lädt.

Das BASIC-Hilfsprogramm ist sicher keine optimale Lösung, aber es ist die bequemste. Man könnte die Aufzeichnung beschleunigen, wenn man die ASCII-Zeichen direkt zum ACIA schreibt (mit POKE), nur wird das Programm dann aufwendiger. Ist die Aufzeichnung fertig, läßt man den Recorder noch etwas weiterlaufen, um Nullzeichen zu erhalten. Dann spult man das Band so zurück, daß der Tonkopf etwa 2 bis 3 s nach der ersten Aufzeichnung steht, denn nun folgt die Eingabe und Aufzeichnung der Texttabelle.

Zu diesem Zweck wird das Hilfsprogramm nach Bild 4b eingegeben und gestartet. Nun kann man die Texttabelle eingeben. Hier sollte man aufpassen, denn ein Rechtschreibfehler kann den Eindruck des Programms auf das Publikum erheblich mindern. Kommt so ein Fehler vor, kann man ihm mit SHIFT 0 verbessern.

Hat man die Tabelle vollständig eingegeben, wird das Hilfsprogramm nach Bild 4c eingegeben, der Recorder eingeschaltet und das Hilfsprogramm ge-

(bitte umblättern)

ELIZA II für 6502-Systeme

Bild 5: Syntax der Texttabelle

```
xkSCHLUSSELWORTtANTWORTSATZ 1fANTWORTSATZ 2fANTWORTSATZ NkSCHLUSSELWORTtALL-
EINSTEHENDER ANTWORTSATZktANTWORT WENN KEIN SCHLUSSELWORT GEFUNDENgKONJUGATION
tERSATZgKONJUGATIONtERSATZbBEGRUSSUNGe
```

Bild 6: Alle Informationen, die nötig sind, um das Programm umzuändern

Adressen:

```
  5Ø Zeiger für Textverarbeitungsroutinen (1 Byte)
  51 Zeiger für Textende (1 Byte)
  52 Länge des Schlüsselwortes (1 Byte)
  53 Nicht benutzt
  54 Zeiger 1 für Texttabelle (2 Byte)
  56 Zeiger 2 für Texttabelle, gibt Antwortsatz an
     (2 Byte)
  58 Nicht benutzt
  59 Hilfsspeicher (1 Byte)
Ø222 Anfang des Textpuffers
Ø2FF Ende des Textpuffers
Ø3ØØ Anfang des Programms
Ø4FF Startadresse
Ø54D Ende des Programms
Ø54E Anfang der Texttabelle
```

Unterprogramme:

```
BF2D Ausgabe eines ASCII-Zei-
     chens (im Accumulator).
     Darf alle Register ver-
     ändern
FEED Eingabe eines ASCII-Zei-
     chens (im Accumulator).
     Darf alle Register ver-
     ändern
```

Spezielle Zeichen:

```
"x"          =78H
"k"          =6BH
"t"          =74H
"f"          =66H
"g"          =67H
"b"          =62H
"e"          =65H
Backspace    =5FH
```

Bild 7: Beispiel einer „Sitzung" mit ELIZA II

```
HI, I'M ELIZA II TELL ME YOUR PROBLEM!
HI
HOW DO YOU DO . . . PLEASE STATE YOUR PROBLEM!
I WANT MONEY TO BUY MORE RAM
WHAT WOULD IT MEAN TO YOU IF YOU GOT MONEY TO BUY MORE RAM
MY COMPUTER WOULD BECOME SMARTER
DO COMPUTERS WORRY YOU?
MAYBE SOMETIMES
YOU DON'T SEEM QUITE CERTAIN!
IN MY OPINION COMPUTERS CAN HELP PEOPLE
ARE YOU TALKING ABOUT ME IN PARTICULAR?
YES
YOU SEEM QUITE POSITIVE.
CAN YOU TELL ME SOMETHING ABOUT YOU?
DON'T YOU BELIEVE THAT I CAN TELL YOU SOMETHING ABOUT ME?
NO, BECAUSE YOU ARE A MACHINE
WHAT MAKES YOU THINK I AM A MACHINE
YOUR CRT AND YOUR KEYBOARD
WHY ARE YOU CONCERNED ABOUT MY CRT AND MY KEYBOARD
THEY PROVE THAT YOU ARE A MACHINE
DOES IT PLEASE YOU TO BELIEVE I AM A MACHINE
YES
ARE YOU SURE?
YES
I SEE
```

startet. Während der Aufzeichnung kann man auf dem Bildschirm mitlesen, was man vorhin eingegeben hat. Sind alle Aufzeichnungen komplett, so kann man den ersten Versuch wagen und die Kassette einlesen (Monitorbefehl L). Man beobachtet, wie zuerst das Hilfsprogramm, dann das Hauptprogramm eingelesen wird.

Schließlich erfolgt ein Start auf der Adresse 0222. Jetzt wird die Texttabelle eingelesen, und nach einiger Zeit sollte die Begrüßungsmeldung erscheinen. Tut sich nichts, so hat man irgendwo einen Eingabefehler gemacht. Wie man sieht, außer dem Ladebefehl ist nichts zu tun, der Vorgang läuft automatisch. Hat alles geklappt, so kann man den ersten Satz eingeben, am besten, man hält sich sofort an die Beispielssätze. Stellt man hier ein anormales Verhalten des Programms fest, so liegt ebenfalls ein Eingabefehler vor.

Vertippt man sich beim Eingeben eines Satzes, so kann man den Fehler mit SHIFT 0 verbessern, der Satz wird mit RETURN abgeschlossen. Bei Vorführungen sollte man es vermeiden, fehlerhafte Sätze einzugeben, denn der Computer wird, falls er Teile des Eingabesatzes weiterverwendet, den gleichen Fehler wieder ausgeben, das Resultat ist dann lautes Gelächter des Publikums.

Besitzt man einen anderen 6502-Computer mit einem besseren Monitor, so kann man ELIZA II als Einheit aufzeichnen, die Startadresse ist 04FF. Die Texttabelle sollte dem Programm unmittelbar folgen, ein paar Byte hin oder her spielen keine Rolle, da sich ELIZA II die Tabelle selber sucht. Das Programm enthält keine Tabellen, es ist also disassemblierbar.

Es läßt sich sicher noch um etwa 10% kürzen, da es auf einfache Korrektur und gute Übersichtlichkeit geschrieben wurde und nicht auf Speicherplatz. Es würde sich aber nicht lohnen, das Programm zu kürzen, denn die gewonnenen 50 bis 60 Byte reichen für etwa drei neue Antwortsätze, was den Aufwand nicht rechtfertigt.

Beschreibung der Funktionsweise

Nach dem Programmstart wird der Arbeitszeiger auf die Mitte der Tabelle gesetzt und von hier an solange um eine Stelle weiterbewegt, bis ein „b" gefunden wurde. Der folgende Text, die Begrüßungsmeldung, wird dann ausgegeben. Nun kann man einen Eingabesatz eintippen, z.B.: „CAN YOU HELP ME?". Nachdem die Eingabe mit „Return" abgeschlossen ist, wird vorne, an der ersten Stelle des Eingabesatzes ein Leerzeichen (␣) eingefügt. Dies muß getan werden, da manche Schlüsselwörter mit einem Leerzeichen beginnen und sonst nicht erkannt werden würden, wenn sie am Anfang des Eingabesatzes stehen. Am Ende des Eingabesatzes wird ein Leerzeichen und das Nullschlüsselwort angefügt. Das Leerzeichen dient, wie oben, zur Schlüsselworterkennung, weil manche Schlüsselwörter mit einem Leerzeichen enden. Das Nullschlüsselwort (†) hat eine besondere Bedeutung. Es leitet in der Texttabelle die letzte Antwortsatzgruppe ein, es wird also nur dann im Eingabesatz gesucht, wenn vorher kein anderes Schlüsselwort gefunden worden ist. Das Resultat ist dann ein allgemeingültiger Antwortsatz, mit dem vertuscht wird, daß der Computer keine eigentliche Antwort gefunden hat.

Nachdem diese Einfügungen gemacht sind, lautet der Eingabesatz wie folgt: „CAN YOU HELP ME?␣†". Nun sucht der Computer seine Texttabelle nach Schlüsselwörtern ab. Er findet sie leicht, weil sie alle mit „k" beginnen. Hat er ein Schlüsselwort gefunden, so sucht er es im Eingabesatz. Findet er es nicht, so wird mit dem nächsten weitergemacht. Findet er es, so sucht er das nächste „t" in der Texttabelle, es markiert dann den nächsten Antwortsatz. In unserem Fall ist das Schlüsselwort „CAN YOU", der Antwortsatz „DON'T YOU BELIEVE THAT I CAN*". Am Ende des Antwortsatzes steht ein Sternchen. Das sagt dem Computer, daß er den Eingabesatz nur bis zum Ende des Schlüsselwortes löschen soll, sonst löscht er ihn vollständig. Übrig ist also „HELP ME␣†". Nun muß der Rest des Eingabesatzes konjugiert werden. Die jeweiligen Konjugationen stehen am Ende der Texttabelle, ein Paar beginnt immer mit „g". Der Computer sucht immer das Wort nach „g" im Rest des Eingabesatzes und ersetzt es, falls es gefunden wird, durch das Wort nach „t". Die Konjugation ist beendet, wenn anstelle eines „g" ein „b", die Begrüßungsmeldung, aufgefunden wird. Der Computer hat also bisher „HELP$*YOU?␣†", den letzten Teil des Antwortsatzes, erzeugt. (Das $ verhindert eine spätere Rückkonjugation.) Nun wird der Antwortsatz aus der Texttabelle vorne eingeschoben. Das Ergebnis ist „DON'T YOU BELIEVE THAT I CAN*HELP$YOU?␣†". Nun brauchen nur noch alle Sternchen, Pfeile und Dollarzeichen in Leerzeichen umgewandelt zu werden, dann kann der fertige Antwortsatz ausgegeben werden. Vorher verschiebt der Computer das „t" so, daß der nächste Antwortsatz des Schlüsselwortes aktiviert wird. Der alte Antwortsatz bekommt dann ein „f" als Steuerzeichen. Ist der letzte Antwortsatz eines Schlüsselwortes verwendet worden, so wird das „t" an den Anfang des ersten Antwortsatzes gestellt, dann wiederholt sich der Computer.

* Das $ verhindert eine spätere Rückkonjugation

Unterprogramme:

0300	INCWP	Erhöht den Arbeitszeiger um 1
0307	DECWP	Erniedrigt den Arbeitszeiger um 1
0315	SRCUP	Stellt den Arbeitszeiger auf das nächste Steuerzeichen, gibt dieses in A zurück.
0321	SRCDW	Stellt den Arbeitszeiger auf das vorige Steuerzeichen, gibt dieses in A zurück.
032D	FNDKW	Sucht das Schlüsselwort, welches ab dem Arbeitszeiger steht, im Eingabesatz. Falls fündig geworden, wird cy gesetzt. Liefert die Länge des Schlüsselwortes auf 52, stellt den Textzeiger (50) auf das erste Zeichen des Schlüsselwortes im Eingabesatz.
035F	CRLFLF	Gibt CR, LF, LF aus.
036F	INSRF	Fügt den ab dem Arbeitszeiger stehenden Text beim Textzeiger in den Eingabesatz ein.
039F	DELW	Löscht den beim Textzeiger stehenden Satzteil, dessen Länge ist vorher auf 59 gespeichert worden.
03B8	DRUTXT	Gibt den Antwortsatz aus.
03CE	INTXT	Eingabe des Eingabesatzes.
03F3	RTWP	Arbeitszeiger retten.
03FC	HLWP	Arbeitszeiger holen.
0405	DELRT	Löscht den Eingabesatz bis zum Ende des Schlüsselwortes.
0417	CHNKSP	Hängt an den Eingabesatz ein Leerzeichen sowie das Nullschlüsselwort an und fügt vorne ein Leerzeichen ein.
0432	CONDL	Wandelt im Ausgabesatz alle Dollarzeichen und Sternchen in Leerzeichen um.
044C	SWPBEG	Setzt den Arbeitszeiger auf den Anfang der nächsten Antwort.
046C	FNDKO	Sucht nächstes Konjugationspaar.
0481	KONJ	Sucht die nächste Konjugationsmöglichkeit im Eingabesatz und konjugiert.
04A0	NTXT	Setzt auf den nächsten Antwortsatz.
04C2	PBEG	Sucht die Texttabelle, setzt Anfangsparameter.
04FF	START	Hauptprogramm.

In kaum einem anderen Bereich lassen sich Programme so effektiv einsetzen wie in der kaufmännischen Anwendung. Oft müssen sie teuer bezahlt werden. Hier einige Programme nicht nur für Kaufleute.
Prozent-, Zins- und Rentenberechnung für Apple II,
PC 1211 hilft bei der Nummernsuche,
Verzinsung mit PC 1211,
Literaturkartei auf dem CBM 2001,
Berechnung der Einkommenssteuer für HP 29 C,
Apple II und Apple II plus als Textsystem.

Gerhard Schischke

Prozent-, Zins- und Rentenberechnung für Apple II

Vorwort zum Programm

Auf sehr vielen Gebieten des täglichen Lebens stößt man auf den Begriff Prozent. Es wird z.B. angegeben, um wieviel Prozent sich in einem bestimmten Zeitraum die Produktion erhöht hat oder die Selbstkosten gesenkt wurden oder wieviel Prozent der Bevölkerung männlichen oder weiblichen Geschlechts ist.
Dies führte zu der Überlegung, ein Programm zu schreiben, das die für den Kaufmann und auch für das tägliche Leben wohl wichtigste Form der Prozentrechnung, die Zinsberechnung, behandelt.
Im Geldverkehr ist es üblich, für eine leihweise Überlassung eines Geldbetrages eine Vergütung zu bezahlen, die von der Höhe des Geldbetrages sowie von dem Zeitraum der Überlassung abhängig ist und Zins genannt wird. Die bis zum Ende eines Zeitraums aufgelaufenen Zinsen werden zum Geldbetrag hinzugerechnet, und im folgenden Zeitraum werden auch von diesen Zinsen wiederum Zinsen berechnet. Man nennt die Berechnung Zinseszinsrechnung. Auch Renten und Tilgungsraten von Anleihen werden mit Zinseszinsen berechnet.
Für die Errechnung von Sparbeträgen, Renten und Tilgungsraten ist daher auf die Zinseszinsformel abzustellen.
Für Sparbeträge gilt dabei folgendes:
Ein Betrag K_0 ergibt zu p Prozent in einem Jahr

$$\frac{K_0 \cdot P}{100}$$

Zinsen. Am Anfang des 2. Jahres wird dann der Betrag

$$K_1 = K_0 + K_0 \cdot \frac{p}{100} =$$

$$= K_0 \left(1 + \frac{p}{100}\right) = K_0\, r$$

verzinst.
Er wächst in einem Jahr um

$$\frac{K_1 \cdot p}{100}$$

an auf

$$K_2 = K_1 + K_1 \frac{p}{100} = K_1\, r = K_0\, r^2$$

Die gleiche Betrachtungsweise gilt für das 3., ⋯ , n-te Jahr. Am Schluß des n-ten Jahres ist der Betrag K_0 durch Zinseszins angewachsen auf

$$K_n = K_0\, r^n$$

wobei

$$r = \left(1 + \frac{p}{100}\right)$$

Aufzinsungsfaktor genannt wird.
Aus der Zinseszinsformel können sowohl die Anzahl n der Jahre als auch der Prozentsatz p berechnet werden. Durch beiderseitiges Logarithmieren erhält man aus der Zinseszinsformel

$$\lg K_n = \lg K_0 + n \lg r$$

oder

$$n = \frac{\lg K_n - \lg K_0}{\lg r}$$

man kann also die Anzahl n der Jahre berechnen.
Durch beiderseitiges Ziehen der n-ten Wurzel ergibt sich dagegen der Aufzinsungsfaktor

$$r = \sqrt[n]{\frac{K_n}{K_0}} = 1 + \frac{p}{100}$$

und aus diesem Prozentsatz

$$p = 100 \left(\sqrt[n]{\frac{K_n}{K_0}} - 1\right)$$

Rentenberechnung

Die wichtigste Form einer Rente ist die Zeitrente, d.h. eine Folge von Zahlungen in voraus festgelegten Zeitpunkten eine bestimmte Anzahl von Zeiträu-

men hindurch. Die einzelnen Zahlungen heißen Raten der Zeitrenten; sie werden entweder am Ende des betrachteten Zeitabschnitts (nachschüssig oder postnumerando) oder seltener am Anfang (vorschüssig oder praenumerando) in vereinbarter Höhe geleistet. Unter dem Endwert einer Zeitrente versteht man den Betrag, auf den die gezahlten Renten am Ende der Laufzeit angewachsen wären, wenn man sie zu p% ausgeliehen hätte. Der Barwert gibt hingegen den Betrag an, der am Anfang der Laufzeit zu zahlen ist, wenn der Endwert der Rente durch eine einmalige Zahlung abgelöst werden soll.

Nachschüssige Zeitrente:
Die am Ende eines Zeitraumes gezahlten Renten b wachsen nach der Zinseszinsformel an. Nach n Zeiträumen hat die 1. Rate den Wert br^{n-1}, die 2. Rate den Wert br^{n-2} usw. Der gesamte Endwert S_n ist danach die Summe einer geometrischen Reihe:

$$S_n = b + br + \cdots + br^{n-1}$$

Daraus folgt:
Der Endwert einer nachschüssigen Zeitrente

$$S_n = b \cdot \frac{r^n - 1}{r - 1}$$

Der Barwert einer nachschüssigen Zeitrente

$$S_n = \frac{b}{r^n} \cdot \frac{r^n - 1}{r - 1}$$

Vorschüssige Zeitrente:
Von einer vorschüssigen Zeitrente wird jede Rate b ein Zeitraum länger verzinst. Nach der Zinseszinsformel ist ihr Endwert S_n danach das r-fache des Endwertes der nachschüssigen Rente. Daraus folgt:
Der Endwert einer vorschüssigen Zeitrente

$$S_n = r\,b \cdot \frac{r^n - 1}{r - 1}$$

Der Barwert einer vorschüssigen Zeitrente

$$S_n = \frac{b}{r^{n-1}} \cdot \frac{r^n - 1}{r - 1}$$

Tilgung einer Anleihe
Die Tilgung von Krediten und Hypotheken wird meist in der Form vorgenommen, daß eine in einem festgelegten Zeitraum zu zahlende, während der Tilgungsdauer unveränderliche Leistung, die Annuität, festgelegt wird. Diese setzt sich aus den Zinsbeträgen und Tilungsbeträgen zusammen.
Die Summe S_n der Tilgungsraten der n ersten Zeiträume entspricht dem Endwert einer nachschüssigen Zeitrente nach n Zeiträumen,

$$S_n = T_1 \cdot \frac{r^n - 1}{r - 1}$$

Aus dieser Gleichung kann man die Anzahl der Zeiträume n berechnen, nach denen die Anleihe getilgt ist. Es gilt:

$$n = \frac{\lg\left(\frac{S}{T_1}(r - 1) + 1\right)}{\lg r}$$

Programmanwendung

Das Programm ist nach dem Start mittels RUN auf Dialogverkehr mit dem Anwender eingerichtet und fragt die erforderlichen Angaben im Menübetrieb selbsttätig ab. Plausibilitätskontrollen werden allerdings nicht durchgeführt, so daß eine gewisse Sorgfalt bei der Eingabe erforderlich ist.
Auch sollten die sich ergebenden Beträge 9,99 Mio. nicht übersteigen, da sonst die auf 10 Stellen eingestellte Routine für 2 Nachkommastellen zu falschen Ergebnissen führt.
Im Bereich der Tilgungen ist noch darauf zu achten, daß bei Eingabe der Annuität eine Tilgung überhaupt möglich ist, da sonst in diesem Programmteil ein Logarithmus mit negativem Vorzeichen entsteht, was zum Programmabbruch führt.
Weiterhin muß, sofern ein Tilgungsplan gewünscht wird, das einzugebende Datum 6 Stellen enthalten (gegebenenfalls auch die führende 0 des Tagesdatums eingeben) und möglich sein. Vom Programm wird allerdings nur das Tagesdatum (1 bis 31) und Monatsdatum (1 bis 12) abgeprüft. Die Eingabe 30.02.80 z.B. ist zulässig. Die Ausgabe über Drucker ist möglich und wird vom Programm während des Menübetriebes jeweils abgefragt. Dabei wird unterstellt, daß der Drucker am Slot 1 angeschlossen ist. Gegebenenfalls ist die Interfacekarte umzustecken oder das Programm entsprechend abzuändern (s. Programmbeschreibung).

Programmbeschreibung

Das Programm ist in Applesoft-BASIC geschrieben und für den Apple II konzipiert. Die Adaption auf andere Systeme sollte aber mit geringfügigen Änderungen möglich sein, da mit Ausnahme der Fehlerbehandlungs- und Druckerroutine kaum spezielle Apple II-Befehle verwendet wurden. Auf diese wird weiter unten noch gesondert hingewiesen. Die Programmlänge beträgt etwa 7K. Während des Programmablaufs werden noch max. weitere knapp 1K Speicherplatz belegt. Das bedeutet, daß bei Systemen mit höchstens 8K Speicherplatz auch bei Weglassen der Fehlerbehandlung und der Druckerroutine (Einsparung etwa 670 Byte) es recht „eng" wird. Schwierigkeiten mag es auch durch die Mehrfachanweisungen je Zeile geben. Nicht jedes System nimmt diese Zeilenlängen anstandslos hin.

Nun zu den einzelnen Programmteilen. Das Hauptprogramm beginnt in Zeile 1000 mit der Initialisierung und verzweigt nach der Programmauswahl in die Subroutinen der Zeilen 1 bis 416. Diese sind wie folgt aufgebaut:

Zeile 1
Ausdruck der Kopfzeile. (Die Bedeutung der einzelnen Strings A$, B$ usw. ist aus den Zeilen 1000 bis 1005 zu ersehen).

Zeile 2
Zeilenvorschub um 3 Zeilen. Bei Verwendung eines Druckers kann es recht nützlich sein, hier eine Variable zu verwenden, die später im Programm den eventuell unterschiedlichen Zeilenvorschub steuert.

Zeile 3
Ausgabe einer gestrichelten Linie auf eine durch die Variable BZ definierte Länge.

Zeile 4
Abfrage der Variablen IN für Programmauswahl.

Zeile 5 bis 8
Abfrage der Varziablen Z für Zeitraum der Zinsabrechnung und Zuordnung des entsprechenden Teilers.

Zeile 9 bis 15
5/4 Rundung und Formatierung auf 2 Nachkommastellen.

Zeile 18 bis 19
Entscheidung, ob Ausgabe über Drukker erfolgen sollte. (Wenn DR = 1, drucken).

(bitte umblättern)

Prozent-, Zins- und Rentenberechnung für Apple II

Zeile 20 bis 39
Bestimmung der einzelnen Variablen.

Zeile 40 bis 46
Formatierung des Datums und Entscheidung, ob der 12. Monat bzw. das 4. Quartal bereits erreicht ist.

Zeile 47 bis 48
Routine für die Zuordnung zu der jeweiligen Tabelle.

Zeile 49
Zähler für die Zwischensummen des Tilgungsplans. T8 = Jahressumme, T9 = Gesamtsumme.

Zeile 50 bis 58
Formatierung der einzelnen Tabellen.

Zeile 60 bis 64
Abfrage, ob Tilgungsplan gewünscht wird.

Zeile 65 bis 69
Abfrage, ob Ausgabe über Drucker erfolgen soll.

Zeile 72 bis 75
Formatierung Tabelle für Tilgungsplan.

Zeile 80 bis 86
Abprüfung, ob Datum möglich und Formatierung des Datums.

Zeile 91
Ausdruck für Tilgungsplan.

Zeile 92 bis 94
Abfrage Taste A, ob Bildschirm neu beschrieben werden kann.

Zeile 110 bis 330
Sprung zur Abfrage der Eingabewerte und Rechenprogramm nach vorstehend beschriebener Zinseszinsformel. Die jeweils benutzte Routine ist aus den Sprungverteilern der Zeilen 1040, 1115, 1125 und 1135 zu ersehen.

Zeile 400 bis 415
Ermittlung der Ausgabewerte für Tilgungsplan.

Zeile 416
Einstellung bei Ausgabedrucker, wann Blattvorschub erfolgen soll. Die Variable GZ erhöht sich mit jeder Print-Anweisung um 1. Im Programm ist die Größe auf ein zufällig vorhandenes Papierformat eingestellt (s. auch Zeile 400).

Für die Adaption auf andere Systeme sind vermutlich folgende Befehle abzuändern:

HOME
Löscht den Bildschirm und bringt den Cursor in die Stellung oben links.

HTAB
Bewegt den Cursor zur Position X (1 bis 40) — horizontal.

VTAB
Bewegt den Cursor auf Spalte X (1 bis 24) — vertikal

PR#1
Output über Drucker, dieser wird durch PR#0 wieder ausgeschaltet.

```
]LIST

0 SZ = 1:Z1 = 1:BZ = 38:NN$ = "?"
    : GOTO 1000
1  HOME : PRINT  TAB( 14)A$: PRINT
    : PRINT B$: RETURN
2  FOR I = 1 TO 3: PRINT : NEXT :
    RETURN
3  FOR I = 1 TO BZ - 1: PRINT "-"
    ;: NEXT : PRINT "-": RETURN

4  PRINT : PRINT "BITTE NUMMER EI
    NGEBEN: ";: GET IN$:IN =  VAL
    (IN$): RETURN
5  PRINT : PRINT W$(Z):BZ =  LEN
    (W$(Z)): GOSUB 3: PRINT : PRINT
    "1 JAEHRL.:": PRINT : PRINT
    "2 QUART. :": PRINT : PRINT
    "3 MONATL.:": PRINT : PRINT
    "BITTE NUMMER EINGEBEN: ";: GET
    Z$:Z =  VAL (Z$): RETURN
6 T = 1: RETURN
7 T = 4: RETURN
8 T = 12: RETURN
9 X =  INT (X * 100 + .5) / 100:X
    $ =  STR$ (X): IF X$ = "0" THEN
    X$ = "0.00": GOTO 15
10  FOR I = 1 TO  LEN (X$): IF  MID$
    (X$,I,1) = "." THEN 12
11  NEXT :X$ =  LEFT$ (X$,I) + ".
    00": GOTO 15
12  IF  LEN (X$) <  = I + 1 THEN
    14
13  FOR J = 1 TO I + 1: NEXT :X$ =
    LEFT$ (X$,J): GOTO 15
14  FOR J = 1 TO I: NEXT :X$ =  LEFT$
    (X$,J) + "0"
15  RETURN
16 ZZ = (ZZ / T) / 100: RETURN
17 ZZ = (ZZ / T) / 100 + 1: RETURN

18  IF DR = 1 THEN  PR# 1: RETURN

19  RETURN
20  GOSUB 1: GOSUB 2: INPUT "ANFA
    NGSBESTAND:";AB: RETURN
21 X = AB: GOSUB 9:AB$ =  RIGHT$
    (X$,10): RETURN
22  PRINT : INPUT "ENDBESTAND
    :";EB: RETURN
23 X = EB: GOSUB 9:EB$ =  RIGHT$
    (X$,10): RETURN
24  PRINT : INPUT "ZINSSATZ
    :";ZZ: RETURN
25 X = ZZ: GOSUB 9:ZZ$ =  RIGHT$
    (X$,10): RETURN
26  PRINT : PRINT "DAUER ";Y$(Z);
    TAB( 15);":";: INPUT JJ: RETURN

27 X = JJ: GOSUB 9:JJ$ =  RIGHT$
    (X$,10): RETURN
28 X = SP: GOSUB 9:SP$ =  RIGHT$
    (X$,10): RETURN
29 X = TT: GOSUB 9:TT$ =  RIGHT$
    (X$,10): RETURN
30  GOSUB 1: GOSUB 2: INPUT "RENT
    ENZAHLUNG :";SP: RETURN
31  PRINT : INPUT "BARWERT
    :";AB: RETURN
32  PRINT : INPUT "ENDWERT
    :";EB: RETURN
```

```
33  PRINT : INPUT "ANNUITAET
    :";SP: RETURN
34  PRINT : INPUT "DARLEHEN
    :";EB: RETURN
35 X = T7: GOSUB 9:T7$ =  RIGHT$
    (X$,10): RETURN
36 X = Z7: GOSUB 9:Z7$ =  RIGHT$
    (X$,10): RETURN
37 X = RS: GOSUB 9:RS$ =  RIGHT$
    (X$,10): RETURN
38 X = T8: GOSUB 9:T8$ =  RIGHT$
    (X$,10):X = Z8: GOSUB 9:Z8$ =
     RIGHT$ (X$,10):X = T9: GOSUB
    9:T9$ =  RIGHT$ (X$,10):X =
    Z9: GOSUB 9:Z9$ =  RIGHT$ (X
    $,10): RETURN
39 X = ZB: GOSUB 9:ZB$ =  RIGHT$
    (X$,10): RETURN
40 DM = DM - 12:DJ = DJ + 1: GOTO
    42
41 DM = DM + MT: IF DM > 12 THEN
     GOSUB 415: GOTO 45
42 DM$ =  STR$ (DM):DJ$ =  STR$ (
    DJ): IF DM < 10 THEN DM$ = "
    0" + DM$
43  IF DJ < 10 THEN DJ$ = "0" + D
    J$
44  RETURN
45  IF DR = 0 THEN  GOSUB 92
46  GOSUB 416: GOTO 40
47  PRINT : GOSUB 55: PRINT : PRINT
    : GOSUB 56: GOSUB 57: RETURN

48  GOSUB 1: GOSUB 2: GOSUB 33: GOSUB
     28: GOSUB 34: GOSUB 23: GOSUB
     24: GOSUB 25: RETURN
49 T8 = T8 + T7:T9 = T9 + T7:Z8 =
    Z8 + Z7:Z9 = Z9 + Z7: RETURN

50  GOSUB 2: GOSUB 1: PRINT : PRINT
    G$; TAB( 20 -  LEN (SP$));SP
    $: PRINT : PRINT S$; TAB( 20
     -  LEN (ZB$));ZB$: RETURN
51  PRINT  TAB( 2);"ANFANGSWERT";
     TAB( 15);"ZINSS."; TAB( 22)
    ;Y$(Z); TAB( 32);L$: GOSUB 3

52  PRINT  TAB( 13 -  LEN (AB$));
    AB$; TAB( 20 -  LEN (ZZ$));Z
    Z$; TAB( 27 -  LEN (JJ$));JJ
    $; TAB( 39 -  LEN (EB$));EB$
    : RETURN
53  GOSUB 2: GOSUB 1: GOSUB 2: PRINT
    "RENTE ";R$(IN): PRINT : PRINT
    : PRINT "ZAHLUNG "; TAB( 20 -
     LEN (SP$));SP$: RETURN
54  PRINT  TAB( 6)"BARWERT"; TAB(
    15);"ZINSS."; TAB( 22);Y$(Z)
    ; TAB( 32);L$: GOSUB 3: RETURN

55  GOSUB 2: GOSUB 1: PRINT : PRINT
    I$,JJ;" ";Y$(Z): PRINT : PRINT
    E$,U$(Z): RETURN
56  PRINT  TAB( 5);T$; TAB( 22);"
    ZINSS."; TAB( 32);"DARLEHEN"
    : GOSUB 3: RETURN
57  PRINT  TAB( 13 -  LEN (SP$));
    SP$; TAB( 27 -  LEN (ZZ$));Z
    Z$; TAB( 39 -  LEN (EB$));EB
    $: RETURN
58  PRINT : PRINT P$(P); TAB( 39 -
     LEN (TT$));TT$: RETURN

60  PRINT : PRINT : PRINT "WIRD T
    ILGUNGSPLAN GEWUENSCHT (J/N)
    ";: GET FR$: IF FR$ = "J" THEN
    63
61  IF FR$ = "N" THEN 99
62  GOTO 60
63 JJ = JJ + 1: GOSUB 1: GOSUB 2:
     INPUT "MANDANT: ";MD$
64  PRINT : INPUT "BEGINN : ";BG$
    : GOSUB 80: GOSUB 86: IF DR =
    1 THEN  RETURN
65  GOSUB 1: GOSUB 2: PRINT "AUSG
    ABE DRUCKER (J/N) ";: GET FR
    $: IF FR$ = "J" THEN DR = 1:
     GOTO 68
66  IF FR$ = "N" THEN DR = 0: RETURN

67  GOTO 65
68  PRINT : PRINT : PRINT "EINSTE
    LL.DRUCKER RICHTIG ?";: PR#
    1: PRINT "*******": PR# 0: PRINT
    : PRINT "WIEDERHOLEN = A : O
    K = B ";: GET FR$: IF FR$ =
    "B" THEN  RETURN
69  GOTO 68
72  GOSUB 1: PRINT : PRINT "TILGU
    NGSPLAN";" ";MD$;: HTAB 33: PRINT
    "SEITE ";SZ
73 BZ = 40: PRINT : PRINT  TAB( 4
    );B$(Z);"/J"; TAB( 12);S$; TAB(
    21);W$(3); TAB( 33);"RESTSCH
    .";: GOSUB 3: RETURN
74  PRINT  TAB( 3);DM$;"/";DJ$; TAB(
    18 -  LEN (Z7$));Z7$; TAB( 2
    8 -  LEN (T7$));T7$; TAB( 40
     -  LEN (RS$));RS$: RETURN
75  GOSUB 3: PRINT "19";DJ$; TAB(
    18 -  LEN (Z8$));Z8$; TAB( 2
    8 -  LEN (T8$));T8$: PRINT "
    GESAMT"; TAB( 18 -  LEN (Z9$
    ));Z9$; TAB( 28 -  LEN (T9$)
    );T9$: RETURN
80  IF  LEN (BG$) <  > 6 THEN 64
81 DT$ =  LEFT$ (BG$,2):DM$ =  MID$
    (BG$,3,2):DJ$ =  RIGHT$ (BG$
    ,2):DT =  VAL (DT$):DM =  VAL
    (DM$):DJ =  VAL (DJ$)
83  IF DT > 31 OR DT < 1 THEN 64
84  IF DM > 12 OR DM < 1 THEN 64
85 BG$ = DT$ + "." + DM$ + "." +
    DJ$: RETURN
86 MT = 12 / T:DM = DM + MT: IF D
    M > 12 THEN DM = DM - 12:DJ =
    DJ + 1
87  GOSUB 42:AD$ = DT$ + "." + DM
    $ + "." + DJ$: RETURN
91  PRINT "TILGUNGSPLAN";" ";MD$:
    BZ = 13 +  LEN (MD$): GOSUB
    3: PRINT : PRINT "DARLEHEN
    ";EB$;"  PER ";BG$: PRINT : PRINT
    T$;" ";U$(Z);"  ";SP$: PRINT
    : PRINT "ERSTMALS AB  ";AD$:
     PRINT : PRINT "ZINSSATZ  ";
    ZZ$;" %";"  LAUFZEIT ";JJ;"
    ";Y$(Z): RETURN
92  PR# 0: VTAB 24: PRINT "TASTE
    'A',WENN WEITER ";: GET FR$:
     IF FR$ = "A" THEN 94
93  POKE 36,0: GOTO 92
94  GOSUB 18: GOSUB 2: RETURN
95  GOSUB 30: GOSUB 28: GOSUB 32:
     GOSUB 23: GOSUB 24: GOSUB 2
```

(bitte umblättern)

Prozent-, Zins- und Rentenberechnung für Apple II

```
    5: GOSUB 18: GOSUB 16:ZF = Z
    Z + 1: RETURN
96  GOSUB 30: GOSUB 28: GOSUB 26:
    GOSUB 24: GOSUB 25: GOSUB 1
    8: GOSUB 17:ZF = ZZ ^ JJ:ZB =
    ZZ:ZF = ZF - 1:ZZ = ZZ - 1: RETURN
97 ZZ = ZZ + 1: FOR X = 1 TO JJ:Z
    1 = Z1 / ZZ: NEXT :AB = EB *
    Z1: GOSUB 21: RETURN
99  PRINT : PRINT  TAB( 14)"*** E
    NDE ***": END
110 GOSUB 20: GOSUB 21: GOSUB 24
    : GOSUB 25: GOSUB 26: GOSUB
    18: GOSUB 17:EB = ZZ ^ JJ *
    AB: GOSUB 23
116 ZB = EB - AB: GOSUB 39: GOSUB
    28: GOSUB 27: RETURN
120 GOSUB 20: GOSUB 21: GOSUB 22
    : GOSUB 23: GOSUB 26: GOSUB
    18:ZV = EB / AB:JZ = 1 / JJ:
    ZZ = ZV ^ JZ * T:ZZ = 100 *
    (ZZ - T): GOSUB 25: GOSUB 11
    6: RETURN
130 GOSUB 20: GOSUB 21: GOSUB 22
    : GOSUB 23: GOSUB 24: GOSUB
    25: GOSUB 18: GOSUB 17:JJ =
    LOG (EB) - LOG (AB):JJ = J
    J / LOG (ZZ): GOSUB 116: RETURN

140 GOSUB 1: GOSUB 2: GOSUB 22: GOSUB
    23: GOSUB 26: GOSUB 24: GOSUB
    25: GOSUB 18: GOSUB 17:ZZ =
    ZZ - 1: GOSUB 97: GOSUB 116:
    RETURN
150 GOSUB 20: GOSUB 21: PRINT : INPUT
    "SPARRATE      :";SP: GOSUB
    28: GOSUB 24: GOSUB 25: GOSUB
    26: GOSUB 16: FOR X = 1 TO J
    J:Z1 = (AB + SP) * ZZ:AB = A
    B + SP + Z1: NEXT :EB = AB: GOSUB
    23
155 EB = VAL (EB$):AB = VAL (AB
    $):SB = AB + ( INT (JJ) * SP
    ):ZB = EB - SB: GOSUB 39: GOSUB
    28: GOSUB 27: RETURN
160 GOSUB 20: GOSUB 21: GOSUB 22
    : GOSUB 24: GOSUB 25: PRINT
    : INPUT "SPARRATE      :";SP
    : GOSUB 28: GOSUB 16:JJ = EB
    / SP: FOR I = 1 TO JJ:Z1 =
    (AB + SP) * ZZ:AB = AB + SP +
    Z1: IF (EB - .01) > AB THEN
    NEXT
165 JJ = I: GOSUB 27:EB = AB: GOSUB
    23:EB = VAL (EB$):JJ = VAL
    (JJ$):AB = VAL (AB$):SB = S
    P * JJ:SB = SB + AB:ZB = EB -
    SB: GOSUB 39: RETURN
210 GOSUB 96:MP = ZB * (ZF / ZZ)
    :EB = MP * SP: GOSUB 23: GOSUB
    27: GOSUB 97: RETURN
220 GOSUB 95:MP = SP * ZF:EB = (
    EB * ZZ) / MP + 1:JJ = LOG
    (EB) / LOG (ZF): GOSUB 27: RETURN
230 GOSUB 96:MP = ZF / ZZ:EB = M
    P * SP: GOSUB 23: GOSUB 97: GOSUB
    27: RETURN
240 GOSUB 95:EB = EB / SP:EB = E
    B * ZZ + 1:JJ = LOG (EB) /
    LOG (ZF): GOSUB 27: RETURN

310 GOSUB 48: GOSUB 18:P = 1: GOSUB
    16:TL = EB * ZZ:T1 = SP - TL
    :S1 = EB / T1:S1 = S1 * ZZ +
    1:ZF = ZZ + 1:TT = LOG (S1)
    / LOG (ZF): GOSUB 29:JJ =
    VAL (TT$): GOSUB 47
315 P = P + 1:ZZ = VAL (ZZ$):JJ =
    INT (JJ): GOSUB 322: GOSUB
    326:TL = VAL (TT$):ZG = ZZ *
    (TL + EB - SP * JJ):TT = EB +
    TL + ZG: GOSUB 29: GOSUB 58:
    P = P + 1:TT = TT / EB:TT =
    (TT - 1) * 100:TT = TT / JJ:
    TT = TT * T: GOSUB 29: GOSUB
    58: RETURN
320 GOSUB 48: GOSUB 26:P = 4: GOSUB
    322: GOSUB 18: GOSUB 325: GOSUB
    327
322 GOSUB 16:TL = EB * ZZ:T1 = S
    P - TL:ZF = (ZZ + 1) ^ JJ:ZF
    = ZF - 1:MP = ZF / ZZ:TT =
    MP * T1: RETURN
325 GOSUB 29: GOSUB 47: GOSUB 58
    :P = P + 1
326 AG = JJ * SP:TT = AG - TT: GOSUB
    29: RETURN
327 GOSUB 58: RETURN
330 GOSUB 48: GOSUB 26:P = 6: GOSUB
    18: GOSUB 17:TL = EB * (ZZ -
    1):T1 = SP - TL:JJ = JJ - 1:
    ZF = ZZ ^ JJ:TT = T1 * ZF: GOSUB
    29:JJ = JJ + 1: GOSUB 47: GOSUB
    58:P = P + 1:TT = SP - TT: GOSUB
    29: GOSUB 58: RETURN
400 ZZ = VAL (ZZ$):ZZ = (ZZ / T)
    / 100:RS = EB:GZ = 7
402 Z7 = RS * ZZ: GOSUB 36:Z7 = VAL
    (Z7$):T7 = SP - Z7: IF RS <
    SP THEN T7 = RS
404 GOSUB 35:T7 = VAL (T7$):RS =
    RS - T7: GOSUB 37:RS = VAL
    (RS$): GOSUB 49: GOSUB 74:GZ
    = GZ + 1: IF RS < .01 THEN
    GOSUB 415: RETURN
406 GOSUB 41: GOTO 402
415 GOSUB 38: GOSUB 75:GZ = GZ +
    2:T8 = 0:Z8 = 0:SZ = SZ + 1:
    RETURN
416 FOR G = 1 TO 24 - GZ: PRINT
    : NEXT : GOSUB 72:GZ = 7: RETURN

1000 A$ = "APPLE COMPUTER":B$ = "
    * PROZENT-,ZINS- UND RENTENB
    ERECHNUNG *":C$ = "SPAREN":D
    $ = "RENTEN":E$ = "TILGUNGEN
    ":F$ = "SPARBETRAG":G$ = "SP
    ARRATE"
1001 H$ = "ZINSSATZ":I$ = "ZEITRA
    UM":J$ = "DISKONTIERUNG":K$ =
    "LFD.SPAREN":L$ = "ENDWERT":
    M$ = "BARWERT":N$ = "NACHSCH
    UESSIGER":O$ = "VORSCHUESSIG
    ER"
1002 Q$ = "SUMME":S$ = "ZINSEN":T
    $ = "ANNUITAET":U$ = "BETRAG
    ":W$(1) = "ZINSABRECHNUNG":W
    $(2) = "RENTENZAHLUNG":W$(3)
    = "TILGUNG":Y$(1) = "JAHRE"
    :Y$(2) = "QUART.":Y$(3) = "M
    ONATE"
1003 R$(1) = "VORSCHUESSIG":R$(2)
    = R$(1):R$(3) = "NACHSCHUES
    SIG":R$(4) = R$(3):U$(1) = "
```

```
      JAEHRL.":U$(2) = "QUART.":U$
      (3) = "MONATL.":8$(1) = "M":
      B$(2) = "Q":8$(3) = "M"
 1005 P$(1) = "LAUFZEIT":P$(2) = "
      ENDSCHULD":P$(3) = "DURCHSCH
      N.VERZINSUNG":P$(4) = "SUMME
       D.TILG.  NACH":P$(5) = "SUM
      ME D.ZINSEN NACH":P$(6) = "T
      ILGUNGSBETRAG IM  ":P$(7) =
      "ZINSBETRAG     IM  "
 1010  ONERR  GOTO 60000
 1030  GOSUB 1: PRINT : PRINT "ES
      KOENNEN FOLGENDE BERECHNUNGE
      N": PRINT : PRINT "DURCHGEFU
      EHRT WERDEN:"
 1040  PRINT : PRINT "1 ";C$: PRINT
      : PRINT "2 ";D$: PRINT : PRINT
      "3 ";E$: GOSUB 4:Z = IN: ON
      IN GOTO 1110,1120,1130
 1110  GOSUB 1: PRINT : PRINT C$:B
      Z =  LEN (C$): GOSUB 3: PRINT
      : PRINT "1 ";F$: PRINT : PRINT
      "2 ";H$: PRINT : PRINT "3 ";
      I$: PRINT : PRINT "4 ";J$: PRINT
      : PRINT "5 ";K$: PRINT : PRINT
      "6 ";I$;" ";K$: GOSUB 4
 1115  GOSUB 65: GOSUB 1: GOSUB 5:
       ON Z GOSUB 6,7,8: ON IN GOSUB
      110,120,130,140,150,160:BZ =
      40: PRINT : GOSUB 50: PRINT
      : PRINT : GOSUB 51: GOSUB 99

 1120  GOSUB 1: PRINT : PRINT D$:B
      Z =  LEN (D$): GOSUB 3: PRINT
      "1 ";M$;" / ";L$: PRINT : PRINT
      "  ";O$;" ";D$: PRINT : PRINT
      "2 ";P$(1): PRINT : PRINT "
       ";O$;" ";D$: PRINT : PRINT
      "3 ";M$;" / ";L$: PRINT : PRINT
      "  ";N$;" ";D$
 1125  PRINT : PRINT "4 ";P$(1): PRINT
      : PRINT "  ";N$;" ";D$: GOSUB
      4: GOSUB 65: GOSUB 1: GOSUB
      5: ON Z GOSUB 6,7,8: ON IN GOSUB
      210,220,230,240:BZ = 40: PRINT
      : GOSUB 53: PRINT : PRINT : GOSUB
      54: GOSUB 52: GOSUB 99
 1130  GOSUB 1: PRINT : PRINT E$:B
      Z =  LEN (E$): GOSUB 3: PRINT
      "1 ";F$(1): PRINT : PRINT "
        ";P$(2): PRINT : PRINT "  "
      ;P$(3)
 1132  PRINT : PRINT "ZWISCHEN";Q$
      ;"N":BZ = 14: GOSUB 3: PRINT
      "2 ";P$(4);" ";NN$;" ";I$: PRINT
      : PRINT "  ";P$(5): PRINT : PRINT
      "3 ";F$(6);" ";NN$;" ";I$: PRINT
      : PRINT "  ";P$(7): GOSUB 4
 1135  GOSUB 65: GOSUB 1: GOSUB 5:
       ON Z GOSUB 6,7,8:BZ = 40: ON
      IN GOSUB 310,320,330: GOSUB
      60: GOSUB 18: GOSUB 2: GOSUB
      2: GOSUB 1: GOSUB 2: GOSUB 9
      1: GOSUB 92: PRINT : PRINT :
       GOSUB 72: GOSUB 400: GOSUB
      99
```

```
 60000 FZ =  PEEK (218) +  PEEK (2
      19) * 256:FC =  PEEK (222): VTAB
      19: PRINT "FEHLER ";FC;" IN
      ZEILE ";FZ
 60002  VTAB 21: PRINT "NEUSTART T
      ASTE 'A',SONST TASTE 'B'";: GET
      FR$: IF FR$ = "A" THEN 0
 60004  IF FR$ = "B" THEN 99
 60005  GOTO 60002
 65535  REM  G.SCHISCHKE (1981)
```

```
BITTE NUMMER EINGEBEN:

              APPLE COMPUTER

* PROZENT-,ZINS- UND RENTENBERECHNUNG *

SPAREN
------

1 SPARBETRAG

2 ZINSSATZ

3 ZEITRAUM

4 DISKONTIERUNG

5 LFD. SPAREN

6 ZEITRAUM LFD. SPAREN

BITTE NUMMER EINGEBEN:
```

(bitte umblättern)

Prozent-, Zins- und Rentenberechnung für Apple II

```
BITTE NUMMER EINGEBEN:

              APPLE COMPUTER

* PROZENT-,ZINS- UND RENTENBERECHNUNG *

RENTEN
------
1 BARWERT / ENDWERT
  VORSCHUESSIGER RENTEN
2 LAUFZEIT
  VORSCHUESSIGER RENTEN
3 BARWERT / ENDWERT
  NACHSCHUESSIGER RENTEN
4 LAUFZEIT
  NACHSCHUESSIGER RENTEN
BITTE NUMMER EINGEBEN:
```

```
BITTE NUMMER EINGEBEN:

              APPLE COMPUTER

* PROZENT-,ZINS- UND RENTENBERECHNUNG *

TILGUNGEN
---------
1 LAUFZEIT
  ENDSCHULD
  DURCHSCHN. VERZINSUNG
ZWISCHENSUMMEN
--------------
2 SUMME D. TILG.   NACH ? ZEITRAUM
  SUMME D. ZINSEN NACH
3 TILGUNGSBETRAG IM   ? ZEITRAUM
  ZINSBETRAG     IM
BITTE NUMMER EINGEBEN:
```

```
                APPLE COMPUTER

* PROZENT-,ZINS- UND RENTENBERECHNUNG *

ZEITRAUM          34.67 MONATE

TILGUNGEN         MONATL.

    ANNUITAET          ZINSS.      DARLEHEN
-------------------------------------------
       400.00           10.00     12000.00

ENDSCHULD                          13866.56

DURCHSCHN. VERZINSUNG                  5.49

WIRD TILGUNGSPLAN GEWUENSCHT (J/N)
```

```
              APPLE COMPUTER

* PROZENT-,ZINS- UND RENTENBERECHNUNG *

TILGUNGSPLAN TEST
-----------------

DARLEHEN  12000.00  PER 15.06.80

ANNUITAET MONATL.   400.00

ERSTMALS AB  15.07.80

ZINSSATZ  10.00 %  LAUFZEIT 35 MONATE
```

```
              APPLE COMPUTER

TILGUNGSPLAN TEST                    SEITE 1

   M/J        ZINSEN     TILGUNG      RESTSCH.
-------------------------------------------------
  07/80        100.00      300.00      11700.00
  08/80         97.50      302.50      11397.50
  09/80         94.98      305.02      11092.48
  10/80         92.44      307.56      10784.92
  11/80         89.87      310.13      10474.79
  12/80         87.29      312.71      10162.08
-------------------------------------------------
1980           562.08     1837.92
GESAMT         562.08     1837.92
```

APPLE COMPUTER

TILGUNGSPLAN TEST SEITE 2

M/J	ZINSEN	TILGUNG	RESTSCH.
01/81	84.68	315.32	9846.76
02/81	82.06	317.94	9528.82
03/81	79.41	320.59	9208.23
04/81	76.74	323.26	8884.97
05/81	74.04	325.96	8559.01
06/81	71.33	328.67	8230.34
07/81	68.59	331.41	7898.93
08/81	65.82	334.18	7564.75
09/81	63.04	336.96	7227.79
10/81	60.23	339.77	6888.02
11/81	57.40	342.60	6545.42
12/81	54.55	345.45	6199.97
1981	837.89	3962.11	
GESAMT	1399.97	5800.03	

APPLE COMPUTER

TILGUNGSPLAN TEST SEITE 3

M/J	ZINSEN	TILGUNG	RESTSCH.
01/82	51.67	348.33	5851.64
02/82	48.76	351.24	5500.40
03/82	45.84	354.16	5146.24
04/82	42.89	357.11	4789.13
05/82	39.91	360.09	4429.04
06/82	36.91	363.09	4065.95
07/82	33.88	366.12	3699.83
08/82	30.83	369.17	3330.66
09/82	27.76	372.24	2958.42
10/82	24.65	375.35	2583.07
11/82	21.53	378.47	2204.60
12/82	18.37	381.63	1822.97
1982	423.00	4377.00	
GESAMT	1822.97	10177.03	

APPLE COMPUTER

TILGUNGSPLAN TEST SEITE 4

M/J	ZINSEN	TILGUNG	RESTSCH.
01/83	15.19	384.81	1438.16
02/83	11.98	388.02	1050.14
03/83	8.75	391.25	658.89
04/83	5.49	394.51	264.38
05/83	2.20	397.80	0.00
1983	43.61	1822.97	
GESAMT	1866.58	12000.00	

*** ENDE ***

Füttern Sie Ihren Apple so, daß er zeigen kann, was in ihm steckt.

EasyWriter ist ein leistungsfähiges Textverarbeitungs-System, entwickelt für jene, die das Beste brauchen.

Mit **Easy Mailer** kann Ihr Apple bzw. ITT 2020 jetzt dazu beitragen, erheblich Zeit, Schreibarbeit und Kosten einzusparen, die Sie bisher für Formbriefe, Adressenetiketten und andere Texte benötigt haben.

EasyMover ist eines der ersten Systeme für eine "Elektronische Post", das die Universalität eines Textverarbeitungs-Systems mit den Möglichkeiten verbindet, Ihre Texte an einen anderen Computer zu senden.

TellStar ist Ihr Computer-Fenster zu den Himmelskörpern, wie sie bei Ihnen zuhause – oder an jedem anderen, von Ihnen gewünschten Punkt am Nachthimmel erscheinen.

SPECIAL SOFTWARE SERVICE

SPECIAL SOFTWARE SERVICE das bedeutet Textverarbeitung, Datei-Management und Kommunikation für Ihren Apple.

Stabelsteiner Weg 22
D-6392 Neu-Anspach 1
Tel.: (06081) 8638
BILDSCHIRMTEXT-Anbieter
Leitseite 5793 — Antwortseite 57903

Software für 8080/8085 – Z 80-Systeme

CP/M Betriebssystem
Konfiguriert für:
TRS 80-Modell I
TRS 80-Modell II
TRS 80-Modell III
APPLE II m. Softkarte
NorthStar
OSI C 3
Heath H 8 + H 89
Zenith Z 89
Mostek MDX
INTEL MDS
ICOM/Pertec
BASF 7100

CP/M-Hard-Disk-Module
für Corvus Disk
S-100-Bus
TRS 80 Mod. II
APPLE II

Programmiersprachen
Basic-80 Interpreter
Basic-80 Compiler
CBasic-2
S-Basic Strukt. Basic Compiler
Fortran-80
Cobol-80
PL/1-80
PASCAL-MT +
C-Compiler
PLMX-80

Datenübertragung
BSTAM, BSTAMS
CP/IBM-3741

Textverarbeitung
Wordstar
WordIndex
MailMerge zum WordStar
MagicWand
NAD (Name + Adresse)

Dateneingabe + Sortierprogramm
DataStar
SuperSort
M/Sort
Q-Sort

Anwendersoftware
Auftragseingang
Lagerhaltung u. Faktur
Kundenkartei
T/MAKER

Entwicklungssoftware
Assembler/Disassembler:
Macro-80
PASM
PLINK
DISTEL (8080)
DISLOG (Z 80)
Z 80-Entwicklungspaket (Editor, Assembl., Linker, Deb.)
APPLE Software Entwicklungspaket (8080/Z 80 6502 Assemb., Linker, Deb.)
Cross Assembler:
XASM-18 für RCA 1802
XASM-48 für MCS 48 u. UPI-41
XASM-65 für 6502 Serie
XASM-68 für 6800
XMACRO-86 für INTEL 8086

NEU
CP/M für APPLE II u. ITT 2020 (Softkarte mit CP/M u. BASIC-80)
WordStar für APPLE II u. ITT 2020
Deutscher Zeichensatz für APPLE II u. ITT 2020
ITT 2020 – Software

Fordern Sie unseren Katalog an!
Diskettenformate: 8" SD (A 1) ab Lager
TRS 80/I-II-III, NorthStar, BASF 7100, Kontron, Superbrain, APPLE II-13/16 Sector, Exidy, Zenith
Ihr LIFEBOAT-Händler in Deutschland

Klaus G. Zimmermann, Ingenieurbüro für Elektronik
Postfach 31 01 70, 6500 Mainz 31, Tel. (0 61 31) 76 96

Ekart Schuberth

PC 1211 hilft bei der Nummernsuche

Beschreibung

Das Programm dient der Speicherung von Telefonnummern, so daß sich der SHARP PC 1211 als Telefonregister verwenden läßt.
Die abgelegten Namen und Nummern werden in zwei getrennte Bereiche aufgeteilt, um den relativ langwierigen Suchlauf zu verkürzen. Ein Bereich könnte private, der andere geschäftliche Namen und Nummern aufnehmen.
Alle Anzeigen wurden bewußt kurzgehalten, um Speicherkapazität zu sparen.
Programmlauf in Modus DEF.

Eingabe:
Bereich 1 — Aufruf mit SHFT =
Bereich 2 — Aufruf mit SHFT M
Nach Aufsuchen einer freien Speicherstelle erscheint Anzeige „MEM-NAME". Nach Eingabe des Namens (max. 7 Stellen) mit ENTER erscheint „MEM-NR". Nach Eingabe der Nummer (max. 11 Stellen) erscheint wieder „MEM-Name" mit der nächsten freien Speicherstelle. Ist im gewählten Bereich kein Speicherplatz mehr vorhanden, so erscheint „VOLL".

Ausgabe:
Bereich 1 — Aufruf mit SHFT „SPC"
Bereich 2 — Aufruf mit SHFT G
Es erscheint Anzeige „NAME?". Nach Eingabe mit ENTER beginnt der Suchlauf. Wird der Name gefunden, erscheint rechtsbündig die Nummer, linksbündig der Name. Wird der Name nicht gefunden, erscheint Anzeige „NOT IN MEM".

Änderungen:
Bereich 1 und 2 gemeinsam mit SHFT L. Es erscheinen nach Druck auf ENTER der Reihe nach alle gespeicherten Namen, gefolgt von „N:". Jetzt kann der zuletzt angezeigte Name überschrieben werden. Es folgt nun Anzeige „NR:". Die dazugehörige Nummer kann nun überschrieben werden. Wird bei „NR:" nichts eingegeben, bleibt die alte Nummer erhalten.

Programm „T-1" (mit Zweiteilung des Speichers)

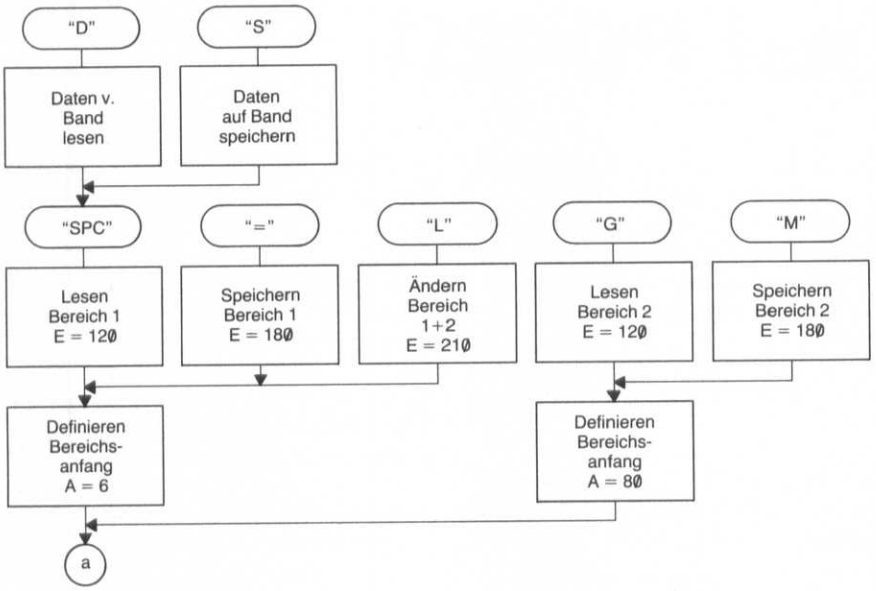
```
 1Ø  "D"   INPUT ↗ "T-1"
 2Ø  " "   E=12Ø:GOTO 8Ø
 3Ø  "G"   E=12Ø:GOTO 9Ø
 4Ø  "S"   PRINT ↗ "T-1":GOTO 2Ø
 5Ø  "="   E=18Ø:GOTO 8Ø
 6Ø  "M"   E=18O:GOTO 9Ø
 7Ø  "L"   E=21Ø
 8Ø        A=6:D=Ø:GOTO 1ØØ
 9Ø        A=8Ø:D=1
1ØØ        BEEP 1
11Ø        FOR A=A TO 146STEP 2:C=A+1:GOTO E
12Ø        INPUT "NAME?",B$:E=E+1Ø
13Ø        IF A$(A)=B$ BEEP 2:PRINT A$(A),A(C):GOTO 16Ø
14Ø        IF A(C)=Ø THEN 23Ø
15Ø        GOTO 22Ø
16Ø        IF A<8Ø THEN 2Ø
17Ø        GOTO 3Ø
18Ø        IF ((D=Ø)*(A>=8Ø))+((D=1)*(A>=148))=1 THEN 23O
19Ø        IF A(C)=Ø INPUT "MEM-NAME",A$(A),"MEM-NR",A(C)
2ØØ        GOTO 22Ø
21Ø        PRINT A$(A):INPUT "N:";A$(A),"NR:";A(C):GOTO 14Ø
22Ø        NEXT A
23Ø        BEEP 3
24O        IF E=13Ø PRINT "NOT IN MEM":GOTO 16O
25Ø        IF E=18Ø PRINT "VOLL"
26Ø        IF E=21Ø PRINT "ENDE"
27Ø        GOTO 2Ø
28Ø        END
```

Programm „T-1" (mit einfacher Speicherbelegung)

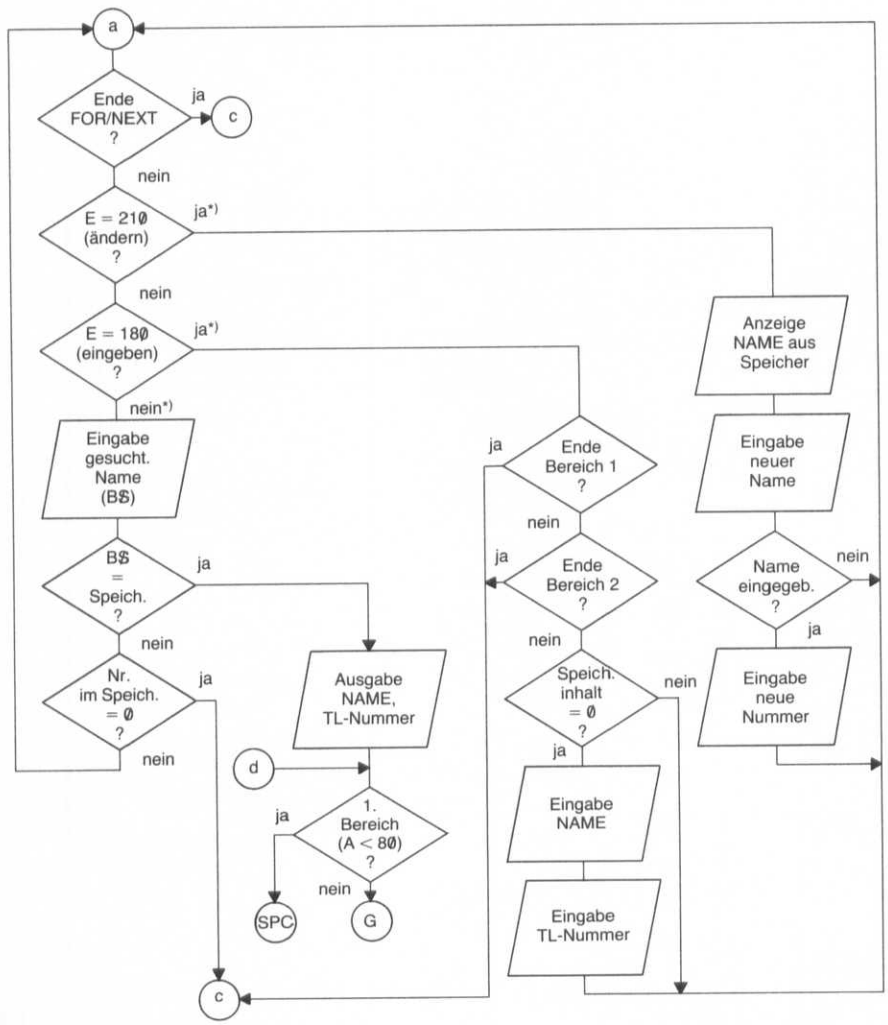
Die folgenden Zeilen sind zu ersetzen:

```
Programm   "T-1"   (mit einfacher Speicherbelegung)
                   Die folgenden Zeilen sind zu ersetzen:
 3Ø  "G"   E=12Ø:GOTO 1ØØ
 6Ø  "M"   E=18Ø:GOTO 1ØØ
11Ø        FOR A=6 TO 158 STEP 2:C=A+1:GOTO E
13Ø        IF A$(A)=B$ BEEP 2:PRINT A$(A),A(C):GOTO 2Ø
18Ø        IF A>158 THEN 23Ø

24Ø        IF E=13Ø PRINT "NOT IN MEM"

Es entfallen die Zeilen: 2Ø, 5Ø, 8Ø, 9Ø, 16Ø, 17Ø
```

Programm „Telefonregister" 1. Teil — EINGABE

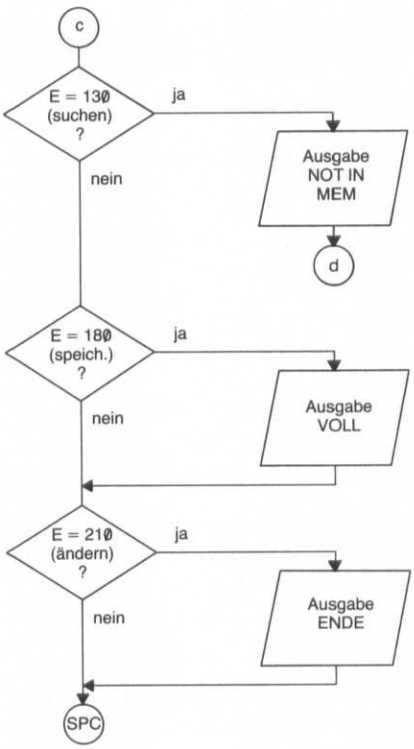


Programm „Telefonregister" 2. Teil — VERARBEITUNG



* Diese Verzweigungen werden erreicht über Zeile 110: GOTO E

Programm „Telefonregister" 3. Teil — BEMERKUNGEN



Wird bei „N:" nichts eingegeben, erscheint der nächste Name.
Bei Erreichen des Speicherendes erscheint „ENDE".
Die FOR-NEXT-Schleife deckt alle Funktionen ab. Arbeitsbeginn innerhalb der Schleife ist die jeweils über den Aufruf festgelegte Zeilennummer (E).
Zeile 160 und 170 führen zum aufgerufenen Datenblock zurück. Zeile 180 prüft auf Speicherende.
SHFT D = Laden der Daten in den PC 1211
SHFT S = Speichern der Daten auf Band.

Version mit einfacher Speicherbelegung

Die Aufrufe sind denen im Bereich 2 gleich. Im Suchlauf macht sich jedoch die langsame Arbeitsweise des PC 1211 noch unangenehmer bemerkbar als bei der unterteilten Version.

Schlußbemerkung

Durch die Unterdrückung führender Nullen und die Möglichkeit, die Vorwahl von der Rufnummer durch einen Punkt zu trennen, ergibt sich die folgende Darstellung der Nummernausgabe:

Anstelle von z.B.	„030/1234567"
Anzeige	„30.1234567"

Heinz Büttner

Verzinsung mit PC 1211

Das Programm ist ausgelegt für die Berechnung von Zins und Zinseszins für positives Kapital.
Das Gesamtprogramm gliedert sich in vier Teile (A, B, C und D), die unter diesen Adressen in die Betriebsart DEFINABLE MODE abgearbeitet werden können.

Der Programmteil A

wird gestartet mit SHFT A. Der Rechner muß in der Betriebsart DEF stehen!
Zunächst erscheint dann in der Anzeige eine stichpunktartige Erläuterung des Programmteiles „B".
Dann zeigt das Display:

VAR. PROG.: 1ENTER

Wollen Sie das Variable Programm (Teil B), so wird die Tastenfolge 1 ENTER gedrückt.
Wird nur die Taste ENTER gedrückt, geht der Rechner zum nächsten Schritt, und es erscheint im Display:

ENDKAP. GES.: 1ENTER

Soll dieser Programmteil abgearbeitet werden, betätigt man analog zum bereits genannten „Variablen Programm" auch hier die Tastenfolge 1 ENTER.
ENTER allein veranlaßt wieder den nächsten Schritt, und es wird angezeigt

STARTKAP. GES.: 1ENTER.

Die Tastenfolge 1 ENTER leitet auch hier die Abarbeitung des angegebenen Programmteils ein.
Die Betätigung der Taste ENTER allein bringt hier jedoch einen Rücksprung zum Satz 5, und es wiederholt sich die Abfrage in der eben erwähnten Reihenfolge.

Der Programmteil B

kann vom Teil A aus, wie beschrieben, gestartet werden. Es besteht jedoch die Möglichkeit, direkt einzusteigen. Dazu wird die Tastenfolge SHFT B betätigt.
In diesem Programmteil, auch „Variables Programm" genannt, kann man z.B. die Vorgänge berechnen, wie sie in einem Sparbuch stattfinden.
So wird ausgehend von einem Starttag, einem Startkapital und dem geltenden Zinssatz bis zu einem definierten Endtag das „Endkapital" errechnet.
Zwischen Start- und tatsächlichem Endtag können Veränderungen in beliebiger Zahl bezüglich Zinssatz und Kapital eingegeben werden.

Es soll anhand eines Beispieles erläutert werden, wie dabei vorzugehen ist:

Kontoeröffnung: 1. Januar 1980
Kapital bei Eröffnung: DM 1000,00
Zinssatz bei Eröffnung: 4,5%

Änderung am: 1. Juli 1980
Kapital plus DM 2000,00 (Einzahlung)
Zinssatz unverändert

Änderung am: 1. Januar 1981
Kapital unverändert
Zinssatz neu mit 5,0%

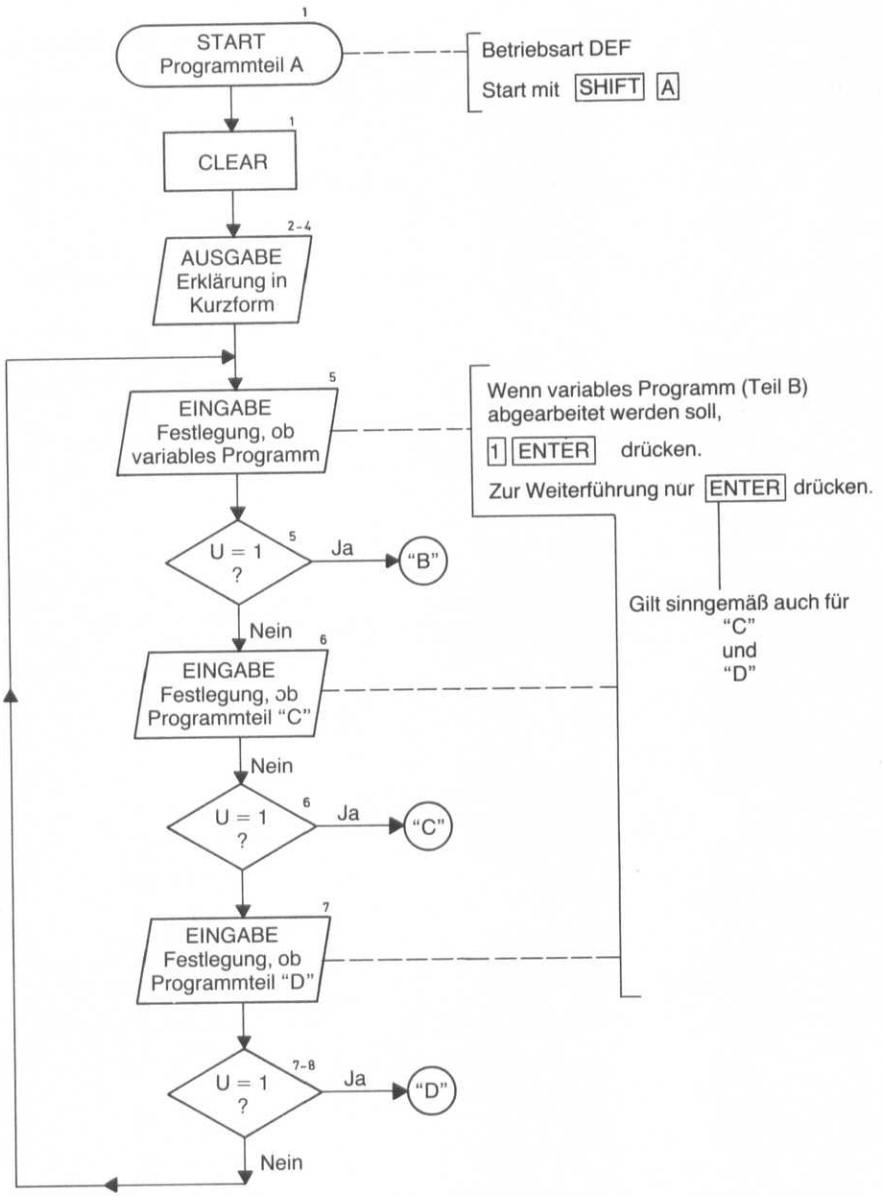
Änderung am: 1. März 1981
Kapital minus DM 500,00 (Abhebung)
Zinssatz neu mit 4,7%

Enddatum tatsächlich: 1. Januar 1982
Kapital an diesem Tag ?????

Arbeitsablauf:

```
EINGABE:   SHFT B          ┐
ANZEIGE:   START-JAHR      │
EINGABE:   1980 ENTER      │ Start-Datum
ANZEIGE:   MONAT           │
EINGABE:   1 ENTER         ┘
ANZEIGE:   TAG
EINGABE:   1 ENTER
ANZEIGE:   END-JAHR        ┐
EINGABE:   1980 ENTER      │
ANZEIGE:   MONAT           │ End-Datum
EINGABE:   7 ENTER         │ ist entweder ein Änderungsdatum,
ANZEIGE:   TAG             │ wie hier,
EINGABE:   1 ENTER         ┘ oder absolutes Enddatum.
ANZEIGE:   STARTKAP.         (Wird entschieden, wenn die Frage
EINGABE:   1000 ENTER        "Änderung?: 1 ENTER"
ANZEIGE:   ZINSSATZ          entweder mit 1 ENTER oder nur
EINGABE:   4.5 ENTER         ENTER beantwortet wird).
ANZEIGE:   AENDERUNG?: 1 ENTER
EINGABE:   1 ENTER
ANZEIGE:   AENDERUNG AM 1.7.1980
                        1.7.1980
           END-Jahr
EINGABE:   1981 ENTER
ANZEIGE:   MONAT
EINGABE:   1 ENTER
ANZEIGE:   TAG
EINGABE:   1 ENTER
ANZEIGE:   KAPITAL +/-
EINGABE:   +2000
ANZEIGE:   ZINSSATZ - - - - - - - Unveränderter Zinssatz,
                                  deshalb keine Eingabe, sondern
                                  nur ENTER zur Weiterführung
EINGABE:   keine; ENTER           des Programmes.
ANZEIGE:   AENDERUNG?: 1 ENTER
EINGABE:   1 ENTER
ANZEIGE:   AENDERUNG AM 1.1.1981
                        1.1.1981
           END-JAHR
EINGABE:   1981 ENTER
ANZEIGE:   MONAT
EINGABE:   3 ENTER
ANZEIGE:   TAG
EINGABE:   1 ENTER
```

```
ANZEIGE:  KAPITAL +/- ----
          Unverändertes Ka-
          pital,    deshalb
          keine Eingabe,
EINGABE:  keine; ENTER son-
          dern nur ENTER zur
          Weiterführung des
          Programmes.
ANZEIGE:  ZINSSATZ
EINGABE:  5 ENTER
ANZEIGE:  AENDERUNG?: 1 EN-
          TER
EINGABE:  1 ENTER
ANZEIGE:  AENDERUNG AM
          1.3.1981
          1.3.1981
          END-JAHR
EINGABE:  1982 ENTER
ANZEIGE:  MONAT
EINGABE:  1 ENTER
ANZEIGE:  TAG
EINGABE:  1 ENTER
ANZEIGE:  KAPITAL +/-
EINGABE:  -500 ENTER
ANZEIGE:  ZINSSATZ
EINGABE:  4.7 ENTER
ANZEIGE:  AENDERUNG?: 1 EN-
          TER
EINGABE:  keine; ENTER
ANZEIGE:  ENDKAPITAL =
          ENDKAPITAL =
          2717.19 DM
EINGABE:  keine; ENTER
ANZEIGE:  E N D E
```



Wie am aufgeführten Beispiel demonstriert, läßt sich eine beliebig lange Kette von Eingaben machen.
Zur genannten Beispielrechnung wurde eine Rechenzeit von ca. zwei Minuten benötigt.
Ergänzend sei erwähnt, daß wie allgemein üblich bei der Berechnung der Tage zwischen dem jeweiligen Start- und End-Datum davon ausgegangen wird, daß jedes Zinsjahr 360 Tage und jeder Zinsmonat einheitlich 30 Tage hat.

Der Programmteil C
geht davon aus, daß für eine Anzahl von ganzen Jahren ein festes Kapital bei einem festen Zinssatz angelegt wird, wie dies z.B. bei sog. abgezinsten Sparbriefen der Fall ist.
Bekannt ist hier das Anfangs- bzw. Startkapital, und der Rechner ermittelt nach Abfrage von Laufzeit und Zinssatz das Endkapital.

Beispiel:
DM 1250,00 werden für 4 Jahre bei einem Zinssatz von 7,5% angelegt. Endkapital = ?????

Arbeitsablauf:

```
EINGABE:  SHFT C
ANZEIGE:  STARTKAP.
EINGABE:  1250 ENTER
ANZEIGE:  LAUFZEIT (JAHRE)
EINGABE:  4 ENTER
ANZEIGE:  ZINSSATZ
EINGABE:  7.5 ENTER
ANZEIGE:  ENDKAPITAL =
          1669.33 DM
EINGABE:  keine; ENTER
ANZEIGE:  E N D E
```

Der Programmteil D
geht wie Teil C ebenfalls davon aus, daß für eine Anzahl von ganzen Jahren ein festes Kapital bei einem festen Zinssatz angelegt wird. Es wird jedoch hier errechnet, welches Startkapital notwendig ist, um nach X Jahren bei einem Zinssatz Y das Endkapital Z zu erhalten.

Beispiel:
Es soll auf die Summe von DM 5000,00 gespart werden. Laufzeit 5 Jahre; Zinssatz 7%. Welches Startkapital wird benötigt?

Arbeitsablauf:

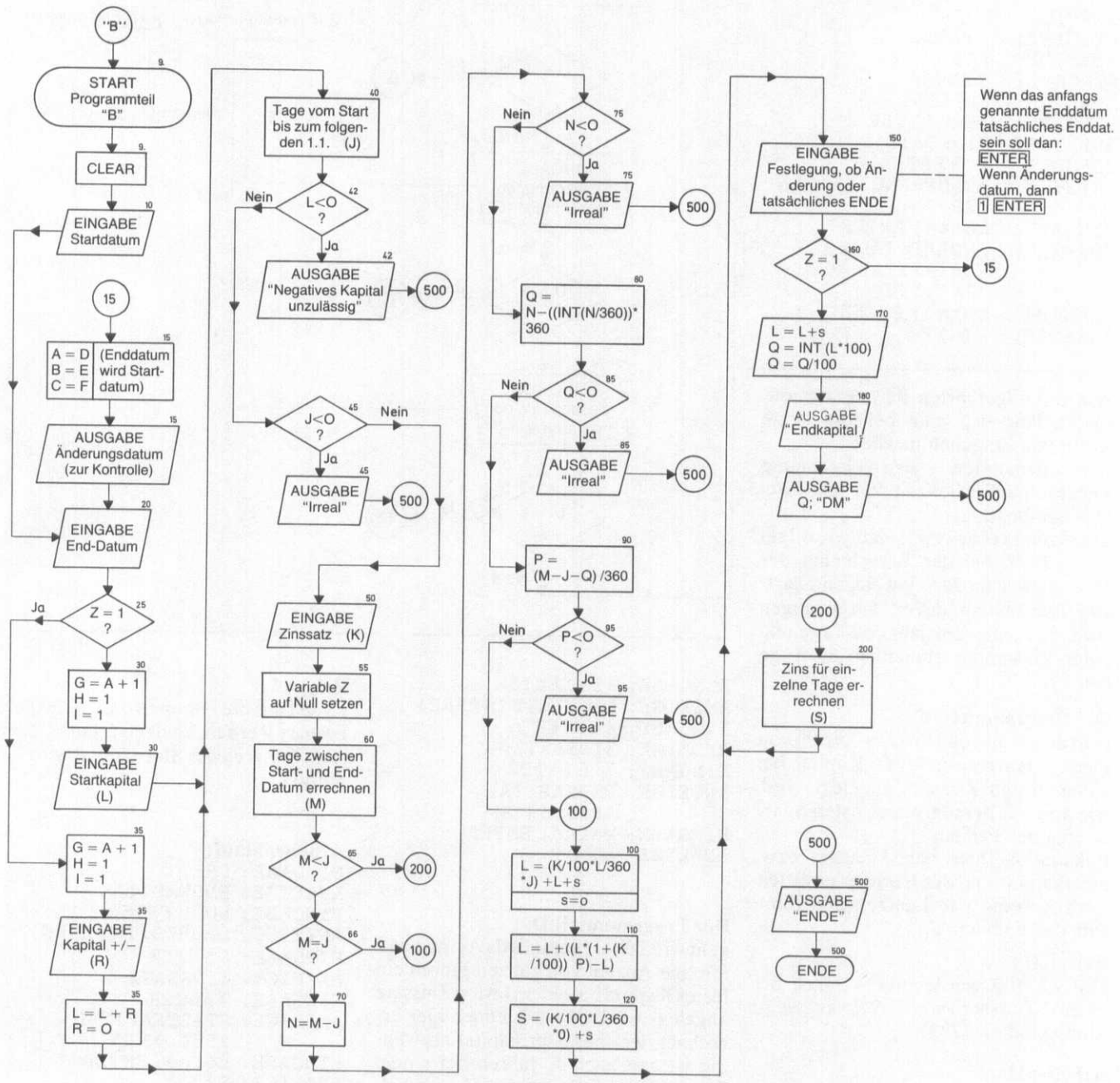
```
EINGABE:  SHFT D
ANZEIGE:  ENDKAPITAL
EINGABE:  5000 ENTER
ANZEIGE:  LAUFZEIT (JAHRE)
EINGABE:  5 ENTER
ANZEIGE:  ZINSSATZ
EINGABE:  7 ENTER
ANZEIGE:  STARTKAPITAL =
          3564.93 DM
EINGABE:  keine; ENTER
ANZEIGE:  E N D E
```

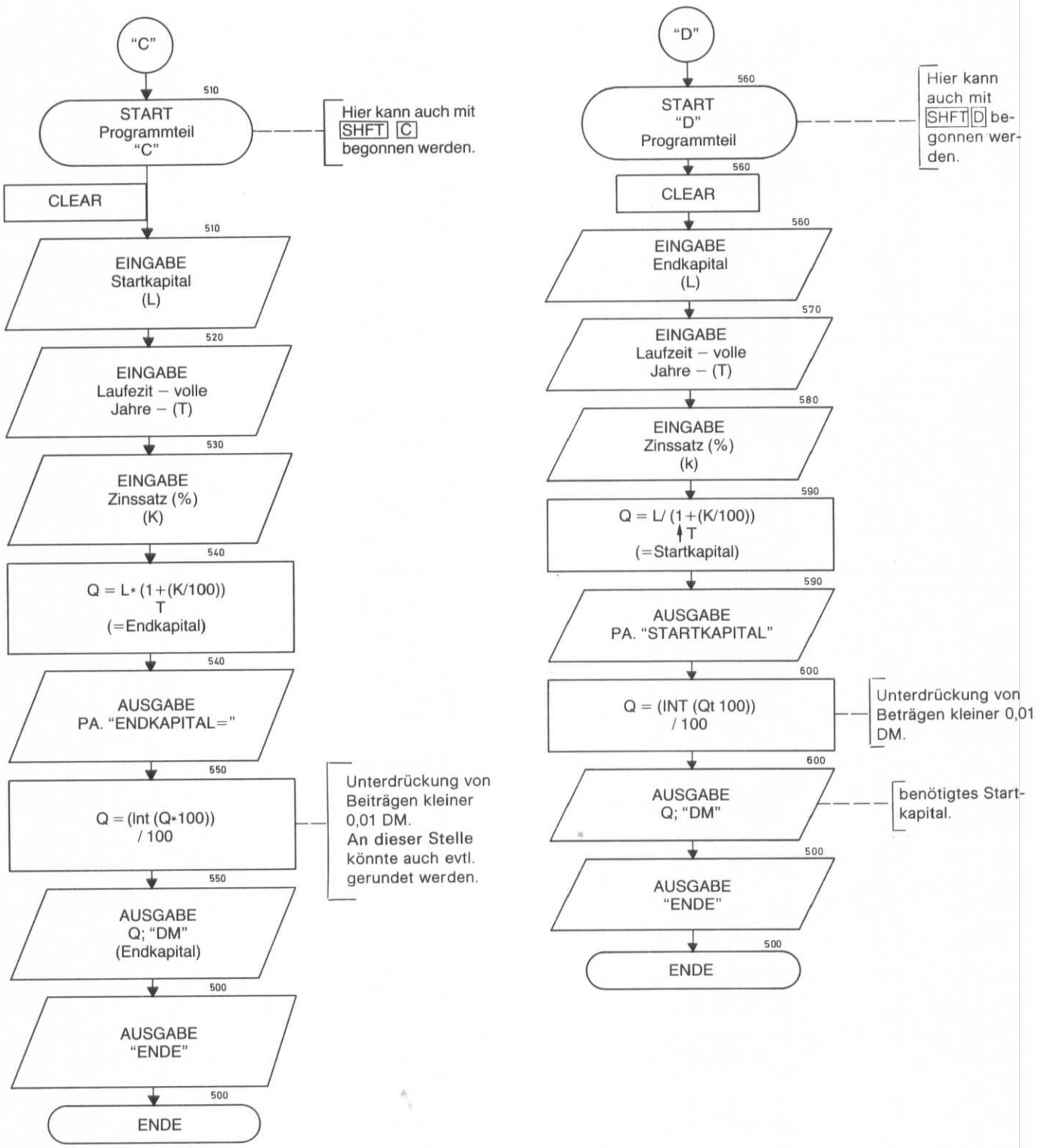
(bitte umblättern)

Verzinsung mit PC 1211

Das errechnete Ergebnis weist einen Betrag aus, bei dem der Anteil kleiner als DM 0,01 unterdrückt wird.
Das Programm läßt sich bei Bedarf jedoch mit kleinem Aufwand dahingehend umstellen, daß der Endbetrag gerundet ausgegeben wird. Möglich wäre beispielsweise die Form, daß man die Stelle der zehntel Pfennige dahingehend überprüft, ob der Wert größer oder gleich fünf ist, oder ob der Wert unter fünf liegt.

Falls Bedingung 1 zutrifft, so wird die Stelle der $^1/_1$ Pfennige um den Wert 1 erhöht; trifft zu, daß der Wert zwischen Null und einem Wert kleiner 5 liegt, so wird die $^1/_1$-Pfennig-Stelle nicht erhöht.





(bitte umblättern)

Verzinsung mit PC 1211

```
LISTING
 1: "A": CLEAR
 2: BEEP 1: PAUSE "VARIABLESØ: PAUSE "PROGRAMM.": PAUSE "ZINSSATZ,": PAUSE "KAPITAL,": PAUSE
    "LAUFZEITØ: PAUSE "KOENNEN"
 3: PAUSE "VARIIERT": PAUSE "WERDEN.": PAUSE " ": BEEP 1: PAUSE "GRUNDKAPITAL": PAUSE "POSI-
    TIV ! !"
 4: PAUSE "1 ZINS-JAHR": PAUSE "= VOM": PAUSE "1. 1. XX": PAUSE "BIS": PAUSE "1. 1. YY":
    BEEP 1
 5: INPUT "VAR. PROG.: 1 ENTER"; U: IF U = 1 GOTO "B"
 6: INPUT "ENDKAP.GES.: 1 ENTER";U: IF U = 1 GOTO "C"
 7: INPUT "STARTKAP.GES.: 1ENTER";U: IF U = 1 GOTO "D"
 8: IF U<> 1 THEN 5

 9: "B": CLEAR
10: INPUT "START-JAHR";A,"MONAT";B,"TAG";C: GOTO 20
15: A=D:B=E:C=F:BEEP2:PAUSE"AENDERUNG":PAUSE"AM":PAUSE C;B;A: PAUSE C;B;A
20: INPUT"END-JAHR";D,"MONAT";E,"TAG";F
25: IF Z=1 THEN 35
30: G=A+1:H=1:I=1:INPUT"STARTKAP. 0";L:GOTO40
35: G=A+1:H=1:I=1:INPUT"KAPITAL +/-";R:L=L+R:R=Ø
40: J=(G-A)*360+(H-B)*30+(I-C)
42: IF L<Ø BEEP3:PRINT"NEG. KAP. UNZULAESSIG":GOTO 500
45: IF J<Ø BEEP3:PRINT"IRREAL":GOTO 500
50: INPUT"ZINSSATZ";K
55: Z=Ø
60: M=(D-A)*360+(E-B)*30+(F-C)
62: IF M<Ø BEEP3:PRINT"IRREAL":GOTO 500
65: IF M<J THEN 200
66: IF M=J THEN 100
70: N=M-J
75: IF N<Ø BEEP3:PRINT"IRREAL":GOTO 500
80: O=N-((INT(N/360))*360)
85: IF O<Ø BEEP3:PRINT"IRREAL":GOTO 500
90: P=(M-J-O)/360
95: IF P<Ø BEEP 3:PRINT"IRREAL":GOTO 500
100: L=(K/100*L/360*J)+L+S:S=Ø
110: L=L+((L*(1+(K/100))↑P)-L)
120: S=(K/100*L/360*O)+S
150: INPUT "AENDERUNG ?: 1ENTER "; Z
160: IF Z=1 THEN 15
170: L=L+S: Q=INT(L*100):Q=Q/100:BEEP3:PAUSE"ENDKAPITAL = ":PAUSE"ENDKAPITAL = "
180: PRINT Q;" DM":GOTO 500
200: S=(K/100*L/360*M)+S:GOTO 150
500: PRINT"- - - - - E N D E":END

510: "C": CLEAR: INPUT"STARTKAP! ";L
520: INPUT"LAUFZEIT (JAHRE) ";T
530: INPUT"ZINSSATZ ";K
540: Q=L*(1+(K/100))↑T:BEEP3:PAUSE"END-KAPITAL ="
550: Q=(INT(Q*100))/100:PRINT Q;" DM": GOTO 500

560: "D":CLEAR:INPUT"ENDKAPITAL ";L
570: INPUT"LAUFZEIT (JAHRE) ";T
580: INPUT"ZINSSATZ ";K
590: Q=L/(1+(K/100))↑T:BEEP3:PAUSE"STARTKAPITAL ="
600: Q=(INT(Q★100))/100:PRINT Q;" DM": GOTO 500
```

Anmerkung zu dieser Auflistung:

An einigen Stellen der Liste wurden der Deutlichkeit wegen Zwischenräume eingefügt.
Wenn alle Zwischenräume beim Eingeben des Programmes in den Rechner PC 1211 berücksichtigt werden, kann es zum Speicherüberlauf kommen. Es empfiehlt sich daher, nur an den entsprechend notwendigen Positionen (INPUT, PAUSE, PRINT), an denen der Rechner etwas anzeigt, die Zwischenräume (SPC) zur Verdeutlichung zu verwenden.
Das Programm selbst benötigt nicht ganz die volle Speicherkapazität von 1424 Schritten. Sollte es dennoch bei der Eingabe zu Fehleranzeige .....4 kommen, d.h. daß der Speicher überfüllt ist, so sind noch an geeigneten Stellen Zwischenräume zu löschen.
Auf den Speicher wirksam sind jedoch nur Zwischenräume innerhalb " ".

SPEICHERBELEGUNG

A Start - Jahr
B Start - Monat
C Start - Tag
D End - Jahr
E End - Monat
F End - Tag
G diese Variable enthält den Wert A+1
H diese Variable enthält den Wert 1
I diese Variable enthält den Wert 1
J Tage vom Start - Datum bis zum nächstfolgenden 1.1.
K Variable für Zinssatz
L Jeweils Kapital (Start-/Zwischenwert)
M Tage zwischen Start- und Enddatum
N Tage M - J
O = M - J - P (Tage danach)
P volle Jahre; Anzeige bei 10 Jahren: 9 (plus J = 360 Tage)
Q Endkapital bei Teil B und C; Startkapital bei Teil D
R diese Variable beinhaltet Kapitalveränderungen
S Zins für einzelne Tage
T Laufzeit in ganzen Jahren (für Programmteil C und D)
U = Sprunganweisung zu "B", "C" oder "D"
V frei
W frei
X frei
Y frei
Z Bedingung für Sprung bei Änderung

Dr. Burkhard Leuschner

Literaturkartei auf dem CBM 2001

Beschreibung

Die Programmserie „Literaturkartei" wurde für den CBM 2001 (8K-Version, mit dem neuen Betriebssystem) geschrieben. Zur Adaption an das alte Betriebssystem — siehe Anhang.
An Peripheriegeräten ist ein Kassettenrecorder notwendig (beim CBM bereits eingebaut). Literaturkartei gibt es inzwischen auch für Floppy-Disk (CBM 3040). In Arbeit ist eine Fassung für den APPLE. Optional (aber sehr hilfreich) ist ein zweiter Kassettenrecorder und ein Drucker. (Das Programm ist eingestellt auf den Drucker ADCOMP X 80.)

Die Programmserie

Die Programmserie besteht aus 2 Teilen:

1. Literaturkartei — Speichern
2. Literaturkartei — Suchen

Ziel der Programmserie

Buchtitel, Titel von Zeitschriftenaufsätzen u.ä. sollen auf Kassette gespeichert und nach Bedarf unter bestimmten Gesichtspunkten wiedergefunden werden.

Die gespeicherten Informationen

Jeder Eintrag enthält die folgenden Informationen:

1. Name des Autors oder Herausgebers
2. Erscheinungsjahr
3. Titel
4. Zeitschrift * Jahrgang (Heftnr.) Seiten
   oder Name der Buchreihe
   oder Angaben zu dem Sammelband, aus dem die Arbeit stammt
5. Ort und Verlag
6. Angaben über Auflage und Erscheinungsjahr, falls notwendig

Dies entspricht dem, was heute in internationalen Veröffentlichungen üblich ist.

Beispiel für einen Buchtitel:

Loeffler, Renate (1979), Spiele im Englischunterricht. Vom lehrergelenkten Lernspiel zum schülerorientierten Rollenspiel, U-&-S-Pädagogik, München: Urban und Schwarzenberg

Beispiel für einen Zeitschriftentitel:

Darian, Steven (1979), The role of redundancy in language and language teaching, System 7 (1) 47-59

Dazu kommen noch drei weitere Kategorien:

7. Angaben zum Standort des Buches oder der Zeitschrift und Signatur, soweit bekannt
8. Bemerkungen (hier kann man z.B. Angaben zum Preis eines Buches machen)
9. Deskriptoren (hier gibt man die Begriffe ein, unter denen man einen Eintrag wiederfinden will).

Die Form der Einträge

Jeder Eintrag ist folgendermaßen aufgebaut:

1. Deskriptoren
2. Name * Vorname
3. Jahr
4. Titel * Untertitel
5. Zs * Jg(H)SS-Reihe-Sammelband
6. Ort * Verlag
7. Auflage * Jahr
8. Standort * Signatur
9. Bemerkungen
10. &

Das Element „&" ist ein Schlußzeichen und gewährleistet, daß bei einem eventuellen Schreib- oder Lesefehler nicht das gesamte Datenfile, sondern nur der betroffene Eintrag fehlerhaft ausgegeben wird.

Die Form der Datenfiles

Die Daten werden in Datenfiles abgelegt. Jedes File hat folgendes Format:

1. Kurzer Einleitungstext, nicht länger als ein 'INPUT' (Datum der Eingabe und ev. Thema der im File enthaltenen Titel, z.B. „1. 1. 80/OELKRISE")
2. ← (Markierungszeichen; es bedeutet, daß danach die eigentlichen Einträge erfolgen)
3. Eintrag
4. Eintrag

...

n. Eintrag

Wieviele Einträge ein Datenfile enthält, hängt ab von der Anzahl der Ein-

(bitte umblättern)

Literaturkartei auf dem CBM 2001

träge, die auf einmal eingegeben werden.
Ein Datenband enthält in der Regel mehrere Files. Erfahrungsgemäß passen mindestens 500 Titel auf eine Seite einer CC90-Kassette.

Speicherprogramm

Das Programm „Literaturkartei — Speichern" dient dazu, die Titel auf Datenband zu schreiben.
Alle Einträge werden grundsätzlich auf 1 geschrieben. Auf Wunsch kann gleichzeitig ein zweites Band und/oder ein Druckerprotokoll hergestellt werden, oder es können einzelne ausgewählte Einträge zusätzlich auf Kass. 2 und/oder Drucker ausgegeben werden.
Alle Eingaben erfolgen über INPUT-Statements. Jede Eingabe darf dabei höchstens 2 Zeilen minus 1 Zeichen lang sein, d.h., der Cursor darf nicht in die 3. Zeile rutschen. Kommas und Doppelpunkte dürfen nicht verwendet werden.
Die Reihenfolge der Eingaben ist frei wählbar; irrelevante Rubriken können automatisch mit „—" gefüllt werden („—" bedeutet: kein Eintrag) — so gibt man z.B. bei Zeitschriftenaufsätzen i.a. keinen Erscheinungsort und Verlag an.
Einige häufig vorkommende Eingabeformate können vorgewählt werden:

1. Normales Format
2. Zeitschriftenhefte
3. Sammelbände
4. Verlagsprospekte
5. Verlagsprospekte (Reihen)
6. Fotokopien

Hinweis: Beim Format ‚Fotokopien' wird vorausgesetzt, daß die gesammelten Kopien durchnumeriert sind. Der Ersteller dieses Programms verschlüsselt jede Fotokopie in der Kategorie ‚Standort' so: LEU*AUFS 138. Die sich immer wiederholende Grundeingabe (LEU*AUFS) wird in Zeile 2290 vorgegeben, die aktuelle Nr. wird dann automatisch daran angehängt. Diese Vorgabe müßte vom Benutzer entsprechend verändert werden.
Nach der grundsätzlichen Entscheidung über das Eingabeformat kann festgelegt werden, daß einige Angaben an der Spitze des Eintrags automatisch wiederholt werden. Dies ist z.B. bei der Auswertung ganzer Zeitschriftenhefte hilfreich, wo man etwa den Zeitschriftentitel nur einmal am Anfang einzugeben braucht. Die Entscheidung über zu wiederholende Eingaben kann nach jedem Eintrag revidiert werden.
Hinweis: Der Ersteller beachtet folgende Eingabekonventionen:
/ trennt Gleichrangiges
* trennt Unterteilungen
— leerer Eintrag
Beispiele siehe unten.
Nach der Eingabe wird jeder Eintrag noch einmal auf dem Bildschirm ausgegeben, und zwar in der Form, in der er dann auf Band gespeichert wird. Er kann jetzt noch geändert werden.
Wenn ein Eintrag auf Band geschrieben ist, wird der Benutzer u.a. gefragt, ob die Eingaben beendet sind. Falls ja, muß die Taste ‚E' gedrückt werden. Es ist absolut erforderlich, diesen Weg zu gehen und die Arbeit mit dem Programm nicht einfach abzubrechen, damit die Datenfiles ordnungsgemäß geschlossen werden. Andernfalls sind sie später nicht lesbar.

Suchprogramm

Das Programm „Literaturkartei — Suchen" dient dazu, die in den Datenfiles gespeicherten Titel zu lesen, wobei je nach Wahl alle im Datenfile enthaltenen Einträge ausgegeben werden können oder nur solche, die nach bestimmten Gesichtspunkten ausgewählt wurden, z.B. um eine Bibliographie zu einem bestimmten Thema herzustellen.
Gesucht werden kann

1. nach einem Einzelbegriff
2. nach mehreren Begriffen gleichzeitig
3. nach mehreren Begriffen alternativ
4. nach einer Kombination aus (2) und (3), d.h. nach mehreren Begriffskombinationen alternativ

Beispiel siehe unten.
Die Anzahl der Begriffe und Begriffskombinationen ist im Prinzip beliebig. Im Zweifelsfall muß die Dimensionierung in Zeile 90 geändert werden: B$(I,J), B%(I,J), Z2(I) — wobei I= die maximale Anzahl der Begriffskombinationen und J= die maximale Anzahl der Begriffe innerhalb einer Kombination.
Die Kategorien, die jeweils durchsucht werden sollen, können frei gewählt werden, meist wird es sich um die Kategorie ‚Deskriptoren' handeln.
Suchbegriffe können abgeblockt werden. Ein Beispiel: Sie suchen alle Einträge, die den Deskriptor „DDR" enthalten. Normalerweise würde auch ein Eintrag mit dem Deskriptor „Schil**ddr**uese" ausgegeben, weil er die Zeichenfolge „DDR" enthält. Dies kann verhindert werden, indem Sie den Suchbegriff abblocken: *DDR*.
Das Programm prüft in diesem Fall nach, ob vor oder nach der Zeichenfolge „DDR" im gefundenen Deskriptor ein Buchstabe steht. Falls ja, dann ist die Suchbedingung nicht erfüllt. Sie ist nur dann erfüllt, wenn dort irgendein anderes Zeichen gefunden wird oder gar keines.
Abblocken empfiehlt sich insbesondere bei Abkürzungen.
Weitere Beispiele:
DIA* (um ‚**Dia**log', ‚**Dia**gramm', ‚**Dia**lekt' auszuschließen)
*LG (um damit ‚Psycho**lg**' auszuschließen)
FU (um damit ‚Ein**fu**ehrung', ‚**fu**er', ‚Kor**fu**' auszuschließen)
(LG ist eine Abkürzung für ‚Linguistik', FU steht für ‚Fremdsprachenunterricht'.)
Die gefundenen Einträge können ausgegeben werden auf dem Bildschirm und/oder auf Kass. 2 und/oder auf dem Drucker. Auf diese Weise ist es z.B. möglich, einen Auszug eines Datenbandes auf einem zweiten Band herzustellen, oder eine Bibliographie zu einem bestimmten Thema ausdrucken zu lassen.
Auf dem Bildschirm werden die Einträge so ausgegeben (**alle** Informationen werden berücksichtigt):

```
 1  . KARTEIEINTRAG                LFD NR 1
-------------------------------------------------------
NAME          MOORWOOD*HELEN(ED)(1978)
TITEL         SELECTIONS FROM MET*MODERN E
NGLISH TEACHER
ZS-REIHE      LONGMAN ENGLISH TEACHING SER
VICES
ORT...        LONDON*LONGMAN      -
STANDORT      -
BEM           ANTH
DESKRIPT      FU/PUZZLE/DISCUSSION/ORAL/IN
TERACTION/GRAMM/PRONUN/COMPR/PROD/AV/GAM
E/TESTING
-------------------------------------------------------
DER EINTRAG SOLL ZUSAETZLICH
2 AUF KASS *2     3 AUSGEDRUCKT WERDEN
ANSONSTEN *IRGENDEINE TASTE DRUECKEN
```

Wie der Bildschirmausdruck zeigt, kann bei Bildschirmausgabe nach Bedarf zusätzlich Ausgabe auf Kass. 2 oder Drucker gewählt werden.

Eine auf dem Drucker erstellte Bibliographie sieht so aus (Deskriptoren, Bemerkungen und leere Einträge werden i.a. nicht berücksichtigt):

```
BIBLIOGRAPHIE

SUCHBEGRIFFE:
  *DIALOG*FU
  *ORAL*INTERACTION*FU
  *DRAMA
  *DISCOURSE ANALYSIS
-------------------------------------------------------------------------

MOORWOOD*HELEN(ED) (1978)
   SELECTIONS FROM MET*MODERN ENGLISH TEACHER
   LONGMAN ENGLISH TEACHING SERVICES
   LONDON*LONGMAN
-------------------------------------------------------------------------

                                                STANDORT: LEU/PH

- (1978)
   PLAYING DETECTIVES TO PRACTISE CONVERSATION
   MET*6(4)6
-------------------------------------------------------------------------

MALEY*A/DUFF*A (1978)
   DRAMA TECHNIQUES IN LANGUAGE
   LONDON*CUP
-------------------------------------------------------------------------

                                                STANDORT: LEU

VIA*RICHARD A (1978)
   THE NOW
   ETF*16(3)24
-------------------------------------------------------------------------

ENDE
```

Anhang

Hinweise für Besitzer des CBM 2001 mit dem alten Betriebssystem

1. Folgende POKE-Befehle sind zu ändern:

PROGRAMM	ZEILE	BEFEHL
LK ERKLAER	180	POKE 245,17
LK SPEICH	120	POKE 245,10
	5040	POKE 245,7
LK SUCH	150	POKE 245,13

2. Folgende Befehle sind einzufügen:

PROGRAMM	ZEILE	STELLE	BEFEHLE
LK SPEICH	4100	nach FOR	POKE 59411,53
		Ende der Zeile	POKE 59411,61
	4250	nach FOR	POKE 59456,207
	4260	Anfang der Zeile	POKE 59456,223
	5000	Anfang der Zeile	POKE 59411,53
		vor CLOSE 2	POKE 59456,207
	8110	vor OPEN	POKE 243,122
			POKE 244,2
	8120	nach OPEN	POKE 59411,53
			POKE 59411,61
	8250	vor OPEN	POKE 243,58
			POKE 244,3
		nach OPEN	POKE 59456,207
	8260	nach PRINT#2	POKE 59456,223
LK SUCH	610	vor OPEN	POKE 243,122
			POKE 244,2
	660	vor OPEN	POKE 243,58
			POKE 244,3
	1210	nach FOR I	POKE 59456,207
		nach PRINT#2	POKE 59456,223

(bitte umblättern)

Hinweise zum Drucker

Der benutzte Drucker ist ein ADCOMP X80.

1. Er wird in den Programmen so angesprochen: OPEN 6, 6, 12.

Dabei ist

6 die logische Filenummer

6 die Gerätenummer (bei den CBM-Druckern entspricht dies der Nr. 4)

12 die Sekundäradresse; sie bewirkt Normaldruck im Grafikmodus

Als Druckbefehle werden benutzt: PRINT#6 und CMD6 mit nachfolgendem einfachen PRINT.

2. Die Cursorzeichen usw. sind im Listing so wiedergegeben:

```
CURSORZEICHEN USW.
------------------

C      CLR BILDSCHIRM

X      CURSOR HOME
       (CURSOR RECHTS)
←      CURSOR LINKS
↓      CURSOR NACH UNTEN
↑      CURSOR NACH OBEN

R      REVERSE ON
Q      REVERSE OFF
```

Literaturkartei auf dem CBM 2001

Speichern

Programmaufbau

```
90      DIM
100     Einleitung
200     Anweisungen betr. Datenbänder und
        Drucker → SBR 8000
300     DATA-Statements (Eingabekatego-
        rien)/
        Versatzstücke für Benutzerinfor-
        mation
400     Wahl des Eingabeformats
 500    Normales Format
 600    Zeitschriftenhefte und Fotokopien
 700    Sammelbände
10800   Verlagsprospekte
1000    Anweisungen betr. die Reihenfolge
        der Eingaben
1100    Anweisungen betr. irrelevante Ein-
        gaben
2000    Eingabeanweisungen betr. Wieder-
        holung
2200    EINGABE DER EINTRAEGE
3000    Überprüfung der Einträge
                  → SBR 3500 bei Änderung
3100    Anweisungen betr. Speicherung des
        Eintrags und Speichern auf Kass 1
                                → SBR 4000
3200    Entscheidungen über Folgeeinträge
3500    Änderungen am Eintrag
4000    SPEICHERN BZW AUSGABE
 4100   Kass 1
 4200   Kass 2
 4600   Drucker
5000    CLOSE Files
7000    Verschiedene kleinere SBR
 7000   Ausgabe der Eingabekategorien
 7100   Berechnung der Reihenfolge bei Än-
        derung in der Reihenfolge
 7200   Betr. Änderungen in der Wiederho-
        lung von Eingaben
 7300   Betr. leere Eingaben
 7400   Benutzerinformation betr. die
        Speichermedien, die benutzt wur-
        den
 7500   Gibt aus, welche Eingaben jeweils
        wiederholt werden
8000    OPEN Files
 8100   Kass 1
 8200   Kass 2
 8600   Drucker
40000   Basis-SBR: Setzt Anführungszei-
        chen bei INPUT-Befehlen
41000   Basis-SBR: Bildschirmausgabe an-
        halten und GET X§-Routine
42000   Basis-SBR: Linie ziehen
```

```
A$
 * *ZWISCHENSPEICHER BASIS VON WIEDERHOLTEN EINGABEN *-
A$(I)
 * *SPEICHERT DIE EINGABEN *Z.B. (BRODBECK*HANS)
B$
 * *ZWISCHENSPEICHER BASIS VON WIEDERHOLTEN EINGABEN *-
C$(I)
 * *SPEICHERT NAMEN DER EINGABEKATEGORIEN *Z.B. (NAME*VORNAME)
C1
 * *ZAEHLER EINTRAEGE DIE AUF KASS #1 GESCHRIEBEN WERDEN *-
C2
 * *ZAEHLER EINTRAEGE DIE AUF KASS #2 GESCHRIFBEN WERDEN *-
C6
 * *ZAEHLER EINTRAEGE DIE AUF DRUCKER AUSGEGEBEN WERDEN *-
CE
 * *ZAEHLER EINTRAEGE *-
E$
 * *VERSATZSTUECK *(EINGABEN)
EI
 * *SPEICHERT ANZAHL IRRELEVANTER ELEMENTE (<-) *-
EI%(I)
 * *SPEICHERT DIE NUMMERN VON IRRELEVANTEN EINGABEELEMENTEN *-
ER
 * *SPEICHERT ANZAHL DER ELEMENTE DIE ANDERE REIHEFOLGE EINNEHMEN *-
ER%(I)
 * *SPEICHERT DEN PLATZ JEDES EINGABEELEMENTS *-
EW
 * *SPEICHERT ANZAHL DER ZU WIEDERHOLENDEN EINGABEN *-
F$
 * *VERSATZSTUECK *(WEITER)
I
 * *SCHLEIFENVARIABLE *-
J
 * *SCHLEIFENVARIABLE *-
J5
 * *SPEICHERT AB WELCHEM PLATZ WIEDERH EINGABEN ZU AENDERN SIND *-
K
 * *SCHLEIFENVARIABLE *-
L$
 * *VERSATZSTUECK *(EINLEGEN)
N
 * *SPEICHERT DIE ANZAHL DER EINGABEN PRO EINTRAG *ZWISCHEN 9
   UND 12
N$
 * *SPEICHERT DATUM (=FILENAME+EINLEITUNGSTEXT KASS #1) *-
NF$
 * *SPEICHERT FILENAME FUER KASS# 2 *-
O2
 * *FLAG EROEFFNETES FILE KASS #2 *1=FILE IST EROEFFNET
O6
 * *FLAG EROEFFNETES DRUCKERFILE *1=FILE IST EROEFFNET
Q
 * *SCHLEIFENVARIABLE *LINIE ZIEHEN
R
 * *FLAG REIHENFOLGE DER EINGABEN *1=NORMAL/2=GEAENDERT
R$
 * *VERSATZSTUECK *(REIHENFOLGE)
S$
 * *VERSATZSTUECK *(SIE HABEN FOLGENDE)
U2
 * *FLAG AUF KASS #2 GESPEICHERTER EINTRAG *1=SCHON GESPEICHERT
UA
 * *FLAG AUSGABE *1=KASS #2/2=DRUCKER/3=KASS #2 + DRUCKER
UD
 * *FLAG AUSGEDRUCKTER EINTRAG *1=EINTRAG SCHON GEDRUCKT
UF
 * *FLAG EINGABEFORMAT *1=NORM./2=ZS/3=SAMMELBD/4=PROSP
/5=BUCHREIH/6=FOTO PEZ
UF$
 * *SPEICHERT EINGABEFORMAT *Z.B. (ZSHEFTE)
UZ
 * *FLAG ZUSAETZLICHE AUSGABE DIE NICHT VORGEWAEHLT
   WURDE *1=KASS #2/2=DR
X$
 * *TEMPORAERE VARIABLE *VGL SBR 41000
W$
 * *VERSATZSTUECK *(WIEDERHOLT WERDEN)

Z$
 * *VERSATZSTUECK *(ANZAHL)
```

Speichern

```
57 REM ROMS 3.0
90 DIMC$(12),A$(12),ER%(12),EI%(5)
100 PRINT"";:FORI=1TO40:PRINT" ";:NEXT
110 PRINT"     L I T E R A T U R K A R T E I";:FORI=1TO46:PRINT" ";:NEXT:PRINT
120 POKE216,10: PRINT"SPEICHERN
130 PRINT"MIT DIESEM PROGRAMM SPEICHERN SIE
140 PRINT"BUCH- UND ZEITSCHRIFTENTITEL AUF BAND.
180 GOSUB42000
190 PRINT"DIE BAENDER KOENNEN MIT DEM PROGRAMM":PRINT,"LITERATURKARTEI- SUCHEN
195 PRINT"AUSGEWERTET WERDEN.":GOSUB42000:GOSUB41000
200 PRINT"ANWEISUNGEN":GOSUB42000
210 PRINT"ALLE EINTRAEGE":PRINT"WERDEN AUF KASS #1 GESCHRIEBEN.
220 PRINT,"0 DABEI BLEIBT'S":PRINT"GLEICHZEITIG SOLL
240 PRINT,"1 EIN 2. BAND AUF KASS #2":PRINT,"2 EIN DRUCKPROTOKOLL
260 PRINT,"3 EIN 2. BAND + DRUCKPROTOKOLL
270 PRINT"HERGESTELLT WERDEN.":GOSUB42000
280 PRINT"BITTE DIE ENTSPRECHENDE ZAHL EINGEBEN.":GOSUB41010:UA=VAL(X$)
290 GOSUB8000
300 DATADESKRIPTOREN,NAME*VORNAME,JAHR,TITEL*UNTERTITEL
310 DATAZS*JG(H)SS-REIHE-SAMMELBAND,ORT*VERLAG,AUFLAGE*JAHR
320 DATASTANDORT*SIGNATUR,BEMERKUNGEN
350 R$=" REIHENFOLGE ":W$=" WIEDERHOLT WERDEN ":Z$="ANZAHL?":E$=" EINGABEN "
360 S$="SIE HABEN FOLGENDE":F$="WEITER":L$=" EINLEGEN
370 FORI=1TOEI:EI%(I)=0:NEXT:EI=0
390 RESTORE:FORI=1TO9:READC$(I):ER%(I)=I:NEXT
400 PRINT"WAHL DES EINGABEFORMATS":GOSUB42000
410 PRINT,"1 NORMALES FORMAT":PRINT,"2 ZEITSCHRIFTENHEFTE
430 PRINT,"3 SAMMELBAENDE":PRINT,"4 VERLAGSPROSPEKTE
450 PRINT,"5 VERLAGSPROSPEKTE (REIHEN)":PRINT,"6 FOTOKOPIEN
460 PRINT,"7 SPEZIELLES FORMAT
470 N=9:UF$="":GOSUB41010:UF=VAL(X$):ONUFGOTO500,600,700,800,800,600,1000
480 GOTO400
500 ER=8:FORI=1TO7:ER%(I)=I+1:NEXT:ER%(8)=1:GOSUB7100:GOTO2000
600 C$(5)="ZEITSCHRIFT":C$(10)="JG(H)":C$(11)="SEITEN":N=11
610 IFUF=6THENC$(12)="NR DER KOPIE":N=12
630 ER=8:ER%(1)=8:ER%(2)=5:ER%(3)=3:ER%(4)=10:ER%(5)=11:ER%(6)=2:ER%(7)=4
650 ER%(8)=1:IFUF=6THENER=9:ER%(5)=12:ER%(6)=11:ER%(7)=2:ER%(8)=4:ER%(9)=1
670 GOSUB7100:EI=2:EI%(1)=9:EI%(2)=10:EW=4:UF$="ZSHEFTE
680 IFUF=6THENEI%(1)=10:EI%(2)=11:UF$="FOTOKOPIE
690 GOTO2000
700 C$(5)="SAMMELBAND (HRSG*KURZTITEL*JAHR)":C$(10)="SEITEN":N=10
710 ER=7:ER%(1)=5:ER%(2)=8:ER%(3)=3:ER%(4)=10:ER%(5)=2:ER%(6)=4:ER%(7)=1
720 GOSUB7100:EI=2:EI%(1)=8:EI%(2)=9:EW=3:UF$="SAMMELBD
730 GOTO2000
800 C$(5)="REIHE":IFUF=5THENC$(10)="BANDNR":N=10
810 ER=6:ER%(1)=3:ER%(2)=6:ER%(3)=5:ER%(4)=2:ER%(5)=4:ER%(6)=1
820 IFUF=5THENER=7:ER%(5)=2:ER%(6)=4:ER%(7)=1:ER%(4)=10
830 GOSUB7100:EI=1:EI%(1)=8:EW=3:UF$="PROSPEKT":IFUF=5THENEI%(1)=9:UF$="REIHE
840 GOTO2000
1000 PRINT"BESTIMMEN SIE BITTE DAS FORMAT, IN DEM  SIE IHRE"E$"MACHEN WOLLEN
1010 GOSUB42000:UF=0
1020 PRINT"WENN SIE KEINE AENDERUNGEN VORNEHMEN,   ERFOLGEN DIE"E$
1030 PRINT"IN DIESER"R$":":GOSUB7000
1040 PRINT"SIE WUENSCHEN":PRINT,"1 DIE OBIGE"R$
1050 PRINT,"2 EINE ANDERE"R$:GOSUB41010:R=VAL(X$):ONRGOTO1100,1060
1060 PRINT"":GOSUB7000:PRINT"WIEVIELE ELEMENTE":PRINT"SOLLEN IHREN PLATZ";
1070 PRINT" AENDERN":PRINTZ$;:GOSUB41010:ER=VAL(X$):PRINTER:IFER=0THEN1100
1080 PRINT"WELCHES ELEMENT AN WELCHEN PLATZ? (NR!)":FORI=1TOER:PRINT"PLATZ"I;
1090 INPUTER%(I):NEXT:GOSUB7100
1100 PRINT"S$;R$"GEWAEHLT":GOSUB42000:GOSUB7000
1120 PRINT"WIEVIELE RUBRIKEN SOLLEN LEER BLEIBEN,  DH AUTOMATISCH MIT <->";
1130 PRINT" GEFUELLT WERDEN":PRINT,Z$;:GOSUB41010:EI=VAL(X$):PRINTEI
1140 IFEI=0THEN2000
1150 FORI=1TOEI:INPUT"NR";EI%(I):NEXT
2000 PRINT"EINGABEANWEISUNGEN",,UF$:GOSUB42000
2030 PRINTS$;R$"GEWAEHLT
2040 GOSUB7000:IFUF>1THENGOSUB41000:GOTO2070
2050 PRINT"WIEVIELE"E$"(AN DER SPITZE DES    EINTRAGS) SOLLEN"W$:PRINT,Z$
2060 GOSUB41010:EW=VAL(X$):IFEW=0THEN2200
2070 PRINT"ES SOLLEN"EW;E$;W$"BITTE EINGEBEN":GOSUB42000
2080 IFJ5THENGOSUB7200
2100 FORI=J5+1TOEW:PRINTC$(ER%(I)):GOSUB40000:INPUTA$(ER%(I)):NEXT
2200 CE=CE+1:PRINT"EINTRAG"CE,,UF$:GOSUB42000:GOSUB7500
2250 FORI=1+EWTON:IFEITHENGOTO7300
2260 PRINT""I;C$(ER%(I))"":GOSUB40000:INPUTA$(ER%(I))
2270 NEXTI:A$=A$(5):B$=A$(8):IFUF=2ORUF=6THENA$(5)=A$(5)+"*"+A$(10)+A$(11)
2280 IFUF=3ORUF=5THENA$(5)=A$(5)+"*"+A$(10)
2290 IFUF=6THENA$(8)=A$(8)+"/LEU*AUFS "+A$(12)
```

(bitte umblättern)

Literaturkartei auf dem CBM 2001

```
3000 PRINT"SIE HABEN EINGEGEBEN":GOSUB42000
3010 PRINTA$(1):PRINT:PRINTA$(2)" ("A$(3)")":PRINTA$(4):PRINT:PRINTA$(5)
3020 PRINTA$(6),A$(7):PRINTA$(8):PRINT:PRINTA$(9):GOSUB42000
3050 PRINT,"AENDERUNGEN A":PRINT,"OK","RETURN":GOSUB41010:IFX$="A"THEN3500
3100 GOSUB4100:IFUATHEN4000
3110 PRINT"DER EINTRAG IST GESPEICHERT":GOSUB7400:GOSUB42000
3130 PRINT"ER SOLL ZUSAETZLICH AUSGEGEBEN WERDEN   AUF
3140 PRINT,"1 KASS #2":PRINT,"2 DRUCKER
3150 PRINTF$:GOSUB41000:UZ=VAL(X$):GOSUB42000:ONUZGOTO4200,4600
3200 U2=0:UD=0:A$(5)=A$:A$(8)=B$:GOSUB7500
3220 IFEWTHENPRINT"NR AB DER DIE ZU WIEDERHOL"E$"ZU     AENDERN SIND
3230 PRINT"F EINGABEFORMAT AENDERN
3240 PRINT"E ENDE DER"E$:PRINTF$:GOSUB41000
3250 J5=VAL(X$):IFEWTHENIFJ5THENJ5=J5-1:GOTO2070
3260 IFX$="F"THEN300
3270 IFX$="E"THEN5000
3280 GOTO2200
3500 PRINT"AENDERUNGEN":GOSUB42000:FORI=1TO9:PRINT""I""A$(I):NEXT:GOSUB42000
3510 PRINT"WELCHE"E$"IST ZU AENDERN? NR":GOSUB41010:A=VAL(X$)
3520 PRINTC$(A):GOSUB40000:INPUTA$(A):GOTO3000
4000 IFUA=3THENUD=0
4010 ONUAGOTO4250,4650,4250
4020 ONUZGOTO4200,4600
4100 C1=C1+1:FORI=1TO9:PRINT#1,A$(I):NEXT:PRINT#1,"&
4110 RETURN
4200 IFO2=0THEN8240
4250 C2=C2+1:FORI=1TO9:PRINT#2,A$(I):NEXT:PRINT#2,"&
4260 U2=1:IFUA=3THENIFUD=0THEN4650
4280 GOTO3110
4600 IFO6=0THEN8650
4650 C6=C6+1:CMD6:IFA$(8)="-"THEN4670
4660 PRINTTAB(50);"STANDORT: ";A$(8)
4670 PRINTA$(2);" (";A$(3);")
4680 PRINT"    ";A$(4):IFA$(5)<>"-"THENPRINT"    ";A$(5)
4700 IFA$(6)<>"-"THENPRINT"    ";A$(6)
4710 IFA$(7)<>"-"THENPRINT"    ";A$(7)
4720 FORQ=1TO80:PRINT"-";:NEXT:PRINT#6:UD=1:GOTO3110
5000 CLOSE1:IFO2=1THENCLOSE2
5020 IFO6=1THENPRINT#6,"ENDE":CLOSE6
5030 PRINT"DIE FILES SIND GESCHLOSSEN
5040 GOSUB42000:POKE216,7:PRINT:PRINT"SCHLUSSBILANZ:
5045 PRINT"EINGEGEBEN WURDEN"CE"EINTRAEGE.
5050 PRINT"FILE AUF","NAME DES FILES"TAB(27)" ANZAHL EINTR":GOSUB42000
5060 PRINT"KASS #1",N$;TAB(27)C1:IFO2THENPRINT"KASS #2",NF$;TAB(27)C2
5080 IFO6THENPRINT"DRUCKER"TAB(27)C6
5090 GOSUB42000:END
7000 PRINT:FORI=1TON:PRINT""I"";C$(ER%(I)):NEXT:PRINT:GOSUB42000:RETURN
7100 FORI=ER+1TON:FORK=NTO1STEP-1:FORJ=1TOI-1:IFK=ER%(J)THEN7130
7110 NEXTJ:ER%(I)=K
7130 NEXTK,I:RETURN
7200 PRINT"BISHER SCHON"W$:FORI=1TOJ5:PRINT""C$(ER%(I))": ";    :
7210 PRINTA$(ER%(I)):NEXT:PRINT"NEU:":RETURN
7300 FORJ=1TOEI:IFER%(I)=ER%(EI%(J))THENA$(ER%(I))="-":GOTO2270
7310 NEXTJ:GOTO2260
7400 PRINT"KASS #1":IFU2=1THENPRINT"KASS #2
7410 IFUD=1THENPRINT"DRUCKER
7420 RETURN
7500 IFEWTHENPRINTW$:FORI=1TOEW:PRINTI;A$(ER%(I)):NEXT:GOSUB42000
7510 RETURN
8000 PRINT"BITTE":PRINT"BAND IN KASS #1"L$
8020 IFUA=1ORUA=3THENPRINT"BAND IN KASS #2"L$
8030 IFUA=2ORUA=3THENPRINT"PAPIER IN DRUCKER"L$
8040 GOSUB42000:PRINT"FERTIG?":GOSUB41000
8100 GOSUB40000:INPUT"DATUM";N$
8110 OPEN1,1,1,N$:PRINT#1,N$:PRINT#1,"_"
8200 IFUA=1ORUA=3THEN8240
8230 GOTO8600
8240 GOSUB40000:INPUT"FILENAME FUER BAND #2";NF$
8250 OPEN2,2,1,NF$
8260 PRINT#2,N$+"/"+NF$:PRINT#2,"_":O2=1:IFUZ=1THEN4250
8600 IFUA=2ORUA=3THEN8650
8640 GOTO8700
8650 OPEN6,6,12:O6=1:CMD6:PRINT"BIBLIOGRAPHIE":PRINT"DRUCKPROTOKOLL: ";N$
8660 FORQ=1TO80:PRINT"-";:NEXT:PRINT#6:IFUZ=2THEN4650
8700 RETURN
40000 POKE158,3:POKE623,34:POKE624,34:POKE625,20:RETURN
41000 PRINT,"*IRGENDEINE TASTE DRUECKEN*
41010 GETX$:IFX$=""THEN41010
41020 RETURN
42000 FORQ=1TO39:PRINT"C";:NEXT:PRINT:RETURN
```

Suchen

```
90      DIM
100     Einleitung
200     Entscheidung über Abruf von Erklärungen          SBR 3000
300     Anweisungen
400     Eingabe Suchbegriffe
500     Anweisung welche Elemente durchsucht werden sollen
600     OPEN Files
  610   Kass 1
  640   Kass 2
  680   Drucker
700     LESEN eines Eintrags von Kass 1
800     Prüfen, ob Suchbegriffe im Eintrag enthalten sind
1000    AUSGABE
  1100  Bildschirm
  1200  Kass 2
  1600  Drucker
1800    Rückkehr zu 700
2000    CLOSE Files
  2100  Entscheidung ob weitere Files gelesen werden sollen
3000    SBR-Erklärungen
5000    SBR: Ständige Benutzerinformation
41000   Basis-SBR: Bildschirmausgabe anhalten
        + GET X$-Routine
42000   Basis-SBR: Linie ziehen
```

Variablen

A$(I) 1. Elemente des Eintrags, der jeweils gelesen wurde (1—10)
2. Temporäre Variable (benutzt in 3000 „Erklärung")

AF *A*nzahl der zu lesenden *F*iles

AK Flag für *A*lte *K*artei (falls AK=1 handelt es sich um eine Kartei des alten Typs; dies ist dann der Fall, wenn der 2. Kartei‚eintrag' nicht ← ist).

AW Flag für die zu berücksichtigenden Einträge: *A*lle: AW-↑
Ausge*w*ählte: AW=2

B$(I,J) Suchbegriffe (I: Nr der jeweiligen Kombination, J: Nr des Begriffs innerhalb der Kombination) (400 und 800)

C Zählt die berücksichtigten Karteieinträge

C$(I) 1. Speichert die Etiketten der einzelnen Elemente des Eintrags (‚Deskriptoren' usw.)
2. Tempöräre Variable (benutzt in 3000 „Erklärung")

C2 Zählt die Einträge im neuen File auf Band 2

CF Zählt die gelesenen Einträge (laufende Nr.)

D2$ Datum/Einleitungstext des neuen Files

DA$ Datum (ev. Einleitungstext) des gelesenen Files

E Anzahl der Elemente, die jeweils zu durchsuchen sind

E%(I) Nr des Elements, die zu durchsuchen sind (1—9) (500,800)

E1$ 2. Element in jedem File; Markierung „←"; weist das Pgm darauf hin, daß es sich um Files des neuen Typs handelt.

I Schleifenvariable

J Schleifenvariable

J1 Schleifenvariable (800)

J2 Schleifenvariable (800)

M$ enthält den jeweils überprüften Stringteil (800)

N Zählt jeweils die Elemente während des Lesevorgangs eines Eintrags (700)

0 Schleifenvariable File1-Eröffnung (600)

02 Flag für offenes File 2 für Kass. 2 (02=1 : File 2 ist offen)

06 Flag für offenes File 6 für Drucker (06=1: File ist offen)

Q Schleifenvariable: Linie ziehen (42000)

UA Flag für *A*usgabeart: UA=1 Bildschirm, UA=2 Kass 2, UA=3 Drucker

UZ Flag für zusätzliche Ausgabe nach Bildschirmausgabe: UZ=2
Kass 2, UZ=3 Drucker (wird nach jeder Bildschirmausgabe neu festgelegt)

Z1 Anzahl der Suchbegriffskombinationen

Z2(I) Anzahl der Suchbegriffe in jeder Kombination

ZB Zählt die eingegebenen Suchbegriffe innerhalb einer Kombination (400)

ZF Zählt die Gesamtfiles, die zu lesen sind

(bitte umblättern)

Literaturkartei auf dem CBM 2001

Suchen

```
READY.
57 REM ROMS 3.0
90 DIMA$(11),B$(11,5),C$(9),Z2(11),B%(11,5),E%(9)
100 PRINT""";:FORI=1TO40:PRINT" ";:NEXT
110 PRINT"      L I T E R A T U R K A R T E I     ";
120 FORQ=1TO40:PRINT" ";:NEXT:PRINT
130 PRINT,,,"    SUCHEN
150 POKE216,13:PRINT:PRINT"MIT HILFE DIESES PROGRAMMS LESEN SIE
160 PRINT"DATENBAENDER, DIE MIT DEM PROGRAMM
170 PRINT,"LITERATURKARTEI - SPEICHERN
180 PRINT"ERSTELLT WORDEN SIND.
190 GOSUB42000
200 GOSUB41000
300 PRINT"ANWEISUNGEN AN DAS PROGRAMM":GOSUB42000
310 PRINT"WIEVIELE FILES":PRINT"SOLLEN HINTEREINANDER GELESEN WERDEN?
320 PRINT,:INPUT"ANZAHL";AF:GOSUB42000:ZF=AF
330 PRINT"WO SOLLEN DIE GEFUNDENEN EINTRAEGE":PRINT"AUSGEGEBEN WERDEN?
340 PRINT"BITTE DIE ENTSPRECHENDE ZAHL EINGEBEN:
350 PRINT,"1 AUF DEM BILDSCHIRM":PRINT,"2 AUF KASSETTE #2
360 PRINT,"3 AUF DEN DRUCKER":GOSUB41010:UA=VAL(X$)
365 IFUA=3THENPRINT"  0 PAPIERROLLE",,"1 DINA4":GOSUB41010:D4=VAL(X$)
370 GOSUB42000:PRINT"WELCHE EINTRAEGE":PRINT"SOLLEN AUSGEGEBEN WERDEN?
380 PRINT,"1 ALLE":PRINT,"2 AUSGEWAEHLTE":GOSUB41010:AW=VAL(X$)
390 ONAWGOTO600,400
400 PRINT"EINGABE DER SUCHBEGIFFE
410 PRINT"LETZTER BEGRIFF JEWEILS: #":PRINT"LETZTE KOMBINATION: @
415 PRINT"ABBLOCKEN EINES BEGRIFSS":PRINT"VORN UND/ODER HINTEN MIT *
420 GOSUB42000
430 Z1=Z1+1:PRINT"BEGRIFFSKOMBINATION"Z1
440 ZB=ZB+1:INPUT"ZZBEGRIFF";B$(Z1,ZB)
450 IFB$(Z1,ZB)="#"THENZ2(Z1)=ZB-1:ZB=0:GOTO400
460 IFB$(Z1,ZB)="@"THENZ1=Z1-1:GOTO480
470 GOTO440
480 FORI=1TOZ1:FORJ=1TOZ2(I)
482 J1=0:J2=0:IFLEFT$(B$(I,J),1)="*"THENJ1=1:B$(I,J)=MID$(B$(I,J),2)
484 IFRIGHT$(B$(I,J),1)="*"THENJ2=1:B$(I,J)=LEFT$(B$(I,J),LEN(B$(I,J))-1)
486 IFJ1THENIFJ2THENB%(I,J)=3:GOTO495
488 IFJ1THENB%(I,J)=1:GOTO495
490 IFJ2THENB%(I,J)=2
495 NEXTJ,I
500 PRINT"JEDER EINTRAG ENTHAELT DIE FOLGENDEN ELEMENTE:":GOSUB42000
510 C$(1)="DESKRIPTOREN":C$(2)="NAME DES AUTORS":C$(3)="ERSCHEINUNGSJAHR
520 C$(4)="TITEL":C$(5)="NAME DER ZEITSCHRIFT/BUCHREIHE":C$(6)="ERSCHEINUNGSORT
530 C$(7)="AUFLAGE":C$(8)="STANDORT":C$(9)="BEMERKUNG
540 FORI=1TO9:PRINT"* "C$(I):NEXT:GOSUB42000
550 PRINT"WIEVIELE ELEMENTE SOLLEN JEWEILS":PRINT"DURCHSUCHT WERDEN?
560 PRINT,"ANZAHL? (ALLE=0)":GOSUB41010:E=VAL(X$):IFE=0THEN595
570 FORI=1TOE:PRINT"SIE WOLLEN"E"ELEMENTE DURCHSEHEN":GOSUB42000
580 PRINT"GEBEN SIE DIE NR DES"I". ELEMENTS EIN:
590 FORJ=1TO9:PRINT""J" "C$(J):NEXT:PRINT,:INPUT"NR";E%(I):NEXT:GOTO600
595 E=9:FORI=1TOE:E%(I)=I:NEXT
600 REM *FILEEROEFFNUNGEN
605 PRINT"BITTE BAND IN KASS #1 EINLEGEN":GOSUB41000
610 FORO=1TOAF:PRINT"ICH SUCHE FILE"ZF-AF+O:OPEN1
620 INPUT#1,DA$:INPUT#1,E1$:GOSUB5000
630 ONUAGOTO700,640,680
640 IFO2=1THEN680
645 PRINT"BITTE BAND IN KASS #2 EINLEGEN":GOSUB41000
650 INPUT"FILENAME FUER BAND #2";NF$:INPUT"DATUM/STICHWORT F. FILEINHALT";D2$
660 OPEN2,2,1,NF$:PRINT#2,D2$:PRINT#2,"_
670 O2=1:GOSUB5000:GOTO695
680 IFO6=1THEN695
685 IFUA=3ORUZ=3THEN690
686 GOTO695
690 OPEN6,6,12:CMD6:PRINT"BIBLIOGRAPHIE":PRINT:O6=1:PG=1:IFZ1=0THEN694
692 GD=GD+1:PRINT"SUCHBEGRIFFE:":FORI=1TOZ1:PRINT"  ";:FORJ=1TOZ2(I)
693 PRINT"*";B$(I,J);:NEXTJ:PRINT:GD=GD+1:NEXTI
694 FORQ=1TO80:PRINT"-";:NEXT:PRINT#6:GD=GD+3
695 IFUZTHEN1000
700 REM* LESEN
710 N=0
720 N=N+1:INPUT#1,A$(N):TS=ST:IFA$(N)="&"THENCF=CF+1:GOTO800
730 IFTS=64THEN2000
740 GOTO720
800 IFAW=1THEN1000:REM*PRUEFEN
810 PRINT""TAB(21)"IN BEARB.: NR "CF
820 FORI=1TOZ1:FORJ=1TOZ2(I):FORJ2=1TOE
```

```
830 IFLEN(A$(E%(J2)))<LEN(B$(I,J))THEN875
840 FORJ1=1TO(LEN(A$(E%(J2)))-LEN(B$(I,J))+1)
850 M$=MID$(A$(E%(J2)),J1,LEN(B$(I,J)))
860 IFM$=B$(I,J)THEN880
870 NEXTJ1
875 NEXTJ2:GOTO890
880 IFB%(I,J)THEN900:REM GEFUNDEN
885 NEXTJ:GOTO1000
890 NEXTI
895 IFTS=64THEN2000
896 GOTO710
900 ZB=B%(I,J)
910 K1=0:IFJ1+LEN(M$)-1<LEN(A$(E%(J2)))THENK1=1
920 IFJ1>1THENIFZB=10RZB=3THENBL(0)=ASC(MID$(A$(E%(J2)),J1-1,1))
930 IFK1THENIFZB=20RZB=3THENBL(1)=ASC(MID$(A$(E%(J2)),J1+LEN(M$),1))
940 IFJ1=1THENBL(0)=0
950 IFK1=0THENBL(1)=0
960 ONZBGOTO970,980,990
970 IFBL(0)<650RBL(0)>90THEN885
975 GOTO870
980 IFBL(1)<650RBL(1)>90THEN885
985 GOTO870
990 IFBL(0)<650RBL(0)>90THENIFBL(1)<650RBL(1)>90THEN885
995 GOTO870
1000 REM*AUSGABE
1010 ONUZGOTO0,1200,1600
1050 C=C+1:ONUAGOTO1100,1200,1600
1100 J=12:PRINT""C". KARTEIEINTRAG"TAB(28)"LFD NR"CF:GOSUB42000:REM*BILDSCH
1110 PRINT"NAME"TAB(J)A$(2)"("A$(3)")
1120 PRINT"TITEL"TAB(J)A$(4):PRINT"ZS-REIHE"TAB(J)A$(5)
1130 PRINT"ORT..."TAB(J)A$(6)"     "A$(7):PRINT"STANDORT"TAB(J)A$(8)
1140 PRINT"BEM"TAB(J)A$(9):PRINT"DESKRIPT"TAB(J)A$(1)
1150 GOSUB42000
1160 PRINT"DER EINTRAG SOLL ZUSAETZLICH
1170 PRINT"2 AUF BAND   3 AUSGEDRUCKT WERDEN
1180 PRINT"ANSONSTEN":GOSUB41000:UZ=VAL(X$):ONUZGOTO0,640,680
1190 GOTO1800
1200 C2=C2+1:A$(10)="&":REM*KASS #2
1210 FORI=1TO10:FORI8=1TO30:NEXT:PRINT#2,A$(I):NEXT
1220 GOTO1800
1600 C6=C6+1:REM*DRUCKEN
1605 IFD4THENIFGD=>53THENGD=1:PRINT#6:PRINT"":GOSUB41000:GOTO1609
1606 GOTO1610
1609 PG=PG+1:PRINT#6,TAB(38);"-"PG"-"
1610 CMD6:GD=GD+1:IFA$(8)="-"THEN1630
1620 PRINTTAB(50);"STANDORT: ";A$(8):GD=GD+1
1630 PRINTA$(2);" (";A$(3);")
1640 PRINT"    ";A$(4):GD=GD+2
1650 IFA$(5)<>"-"THENPRINT"    ";A$(5):GD=GD+1
1660 IFA$(6)<>"-"THENPRINT"    ";A$(6):GD=GD+1
1670 IFA$(7)<>"-"THENPRINT"    ";A$(7):GD=GD+1
1680 FORQ=1TO80:PRINT"-";:NEXT:PRINT#6:GD=GD+1
1800 UZ=0:GOSUB5000
1805 IFTS=64THEN2000
1810 GOTO710
2000 REM*FILES SCHLIESSEN
2010 CLOSE1:NEXT0
2020 PRINT"SOLLEN WEITERE FILES GELESEN WERDEN? J/N":GOSUB41010
2030 IFX$="J"THEN2100
2040 IF02=1THEN:FORI8=1TO70:NEXT:CLOSE2
2050 PRINT"GEPRUEFT WURDEN"CF"EINTRAEGE":PRINT"IN"ZF"FILES.
2060 PRINT"GEFUNDEN WURDEN"C"EINTRAEGE.
2070 IF02THENPRINT"DAS NEUE FILE ENTHAELT"C2"EINTRAEGE.
2080 IF06=1THENPRINT#6,"ENDE":CLOSE6:PRINT"AUSGEDRUCKT WURDEN"C6"EINTRAEGE.
2090 END
2100 PRINT"EVENTUELL NEUES BAND IN KASS #1 EINLEGEN
2110 PRINT"DANN:":GOSUB41000:GOSUB42000:INPUT"ANZAHL ZU LESENDE FILES";AF
2120 ZF=ZF+AF:GOTO610
5000 REM*LFD BENUTZERINFORMATION
5010 PRINT"FILE"ZF-AF+0"(VON"ZF")":PRINTDA$:PRINT"ICH SUCHE KARTEIEINTRAG"C+1
5020 PRINT:IFZ1=0THENPRINT"AUSGEGEBEN WERDEN ALLE EINTRAEGE":RETURN
5030 PRINT"*SUCHBEGRIFFE:":FORI=1TOZ1:FORJ=1TOZ2(I):PRINTB$(I,J)+"+";:NEXTJ
5040 PRINT" ":NEXTI
5050 PRINT"*DURCHSUCHT WERDEN:":FORI=1TOE:PRINTC$(E%(I)):NEXT:RETURN
41000 PRINT,"*IRGENDEINE TASTE DRUECKEN*
41010 GETX$:IFX$=""THEN41010
41020 RETURN
42000 FORQ=1TO39:PRINT"C";:NEXT:PRINT:RETURN
READY.
```

Dr. Heinrich Schmidt

Berechnung der Einkommensteuer für HP 29C

Beschreibung

Als geplagter Steuerzahler habe ich mich immer wieder gewundert, wie gewaltig der Fiskus zuschlägt und wollte endlich einmal wissen, ob das auch stimmt, was „die" einem abziehen. Nun gut, man kann Steuertabellen kaufen und nachschlagen. Im Zeitalter des Mikrocomputers ist dieses Vorgehen für einen einigermaßen mathematisch gebildeten Zeitgenossen jedoch schlichtweg unakzeptabel. Ich habe mir deshalb die Mühe gemacht, meinen HP 29C dahin zu bringen, diese mühsame Nachschlagearbeit überflüssig zu machen; m.a.W., ich habe ein Programm gemacht, das die Steuerschuld aus dem zu versteuernden Jahreseinkommen berechnet, und zwar sowohl für die Einkommensteuer-Grund- als auch die Splittingtabelle (d.h. gültig für Ledige und Ehepaare). Das Programm ist universell insofern, als es jeden der verschiedenen Einkommensbereiche umfaßt, in denen unterschiedliche Steuerformeln gelten.

Eines muß man klar sagen, das Programm ersetzt natürlich keinen Steuerberater, denn Eingangsgröße ist das zu versteuernde Einkommen — das ist das steuerpflichtige Einkommen abzüglich individuell verschiedener Freibeträge — und bekanntlich besteht die Kunst des Steuersparens darin, möglichst viel „abzusetzen". Trotzdem habe ich mein Programm in den letzten zwei Jahren schon oft nutzbringend anwenden können. Man kann es z.B. dahingehend modifizieren, daß es aus dem jeweiligen Monatseinkommen die monatlichen Steuerabzüge berechnet bzw. das zu erwartende Nettomonatsgehalt. Oder man kann die Steuerersparnis im voraus berechnen, die sich aus einer absetzungsfähigen Investition ergibt (Wohnungs- oder Hauskauf oder dergleichen).

Berechnung der Steuerschuld nach der Einkommensteuer-Grund- und Splittingtabelle:

Bezeichnungen und Definitionen:

E...zu versteuerndes Jahreseinkommen in DM (= steuerpflichtiges Einkommen abzüglich individuell verschiedener Freibeträge)

$S_{GT}(E)$... Jahressteuerschuld nach der Grundtabelle in DM

$S_{ST}(E)$... Jahressteuerschuld nach der Splittingtabelle in DM

Hilfsvariable (laut EStG):

1) $\varepsilon = \varepsilon(E) = \dot{Q}\ \mathrm{Int}(E/Q^{1})$

wobei gilt

für 1980: $Q = Q(E) = \begin{cases} 30 \text{ für } E < E_3^{2)} \\ 60 \text{ für } E \geq E_3 \end{cases}$

und für 1981: $Q = 54$ für alle E

Dies bedeutet eine Rundung des Einkommens auf ganzzahlige Vielfache von 30 bzw. 60 DM (für 1980) oder 54 DM (für 1981).

2) $y = y(E) = (\varepsilon - \varepsilon_1)/10\,000$ für $E_2 \leq E < E_3^{2)}$

3) $z = z(E) = (\varepsilon - \varepsilon_2)/10\,000$ für $E_3 \leq E < E_4$

4) *Hilfsfunktion:*

$I(x) = \mathrm{Int}(1000x)/1000$

I(x) schneidet die vierte und höhere Nachkommastellen einer Zahl x ab.

Mit diesen Definitionen lauten die **Steuerformeln** (laut EStG)

a) für die **Grundtabelle**[2)]

$$S_{GT}/E) = \begin{cases} S_1(E) = 0; & 0 \leq E < E_1 \\ S_2(E) = \mathrm{Int}(I(a_1\varepsilon) + a_0); & E_1 \leq E < E_2 \\ S_3(E) = \mathrm{Int}(I((((I((b_4y)+b_3)y)+b_2)y)+b_1)y)+b_0); & \\ \qquad E_2 \leq E < E_3 & \\ S_4(E) = \mathrm{Int}(I((((I((c_4z)+c_3)z)+c_2)z)+c_1)z)+c_0); & \\ \qquad E_3 \leq E < E_4 & \\ S_5(E) = \mathrm{Int}(I(d_1\varepsilon)+d_0); & E_4 \leq E < \infty \end{cases}$$

b) für die **Splittingtabelle:**

$S_{ST}(E) = 2S_{GT}(E/2); \quad 0 \leq E < \infty$

d.h., die Steuerschuld nach der Splittingtabelle ergibt sich als das Doppelte dessen, was sich aus der Hälfte des Einkommens (beider Ehepartner) nach der Grundtabelle ergibt.

Die obige Formel für $S_{GT}(E)$ sieht kompliziert aus; im Grunde jedoch handelt es sich dabei nur um Polynomberechnungen nach Maßgabe gewisser Rundungsvorschriften, die der Gesetzgeber vorschreibt (siehe EStG). Ohne Beachtung dieser Rundungsvorschriften würde z.B. die Steuerformel für den Einkommensbereich $E_1 \leq E < E_2$ (1981 ist das der Bereich zwischen 4213 und 18 000 DM) lauten:

$S_2(E) = a_1E + a_0$

1) Die Funktion Int(x) schneidet die Nachkommastellen einer Zahl x ab (z.B. Int(15.3456) = 15)

2) Wegen der Konstanten E_i, ε_i, a_i,... etc. siehe Tabelle 1

Dies ist eine einfache lineare Funktion des Einkommens E. Das Verhältnis von Steuerzuwachs zu Einkommenszuwachs ist in diesem linearen Bereich konstant, und zwar gleich $a_1 = 0.22$ (Steuersatz 22%). Im darauffolgenden Einkommensbereich $E_2 \leq E < E_3$ (1981: 18 001 bis 59 999 DM) würde gelten:

$$S_3(E) = (((b_4y + b_3)y + b_2)y + b_1)y + b_0$$

$$= b_4y^4 + b_3y^3 + b_2y^2 + b_1y + b_0$$

Hier handelt es sich um ein Polynom 4. Grades in y, das nach dem Horner-Schema berechnet wird. Dies gilt analog auch für $S_4(E)$ im nächstfolgenden Einkommensbereich. In diesem „progressiven" Bereich $E_2 \leq E < E_4$ (1981: 18 001 bis 129 999 DM) nimmt der „Steuersatz" „stetig" zu, bis er schließlich im wiederum linearen Bereich $E_4 \leq E < \infty$ (1981: über 130 000 DM) den Maximalwert $d_1 = 0.56$ (56%) erreicht.

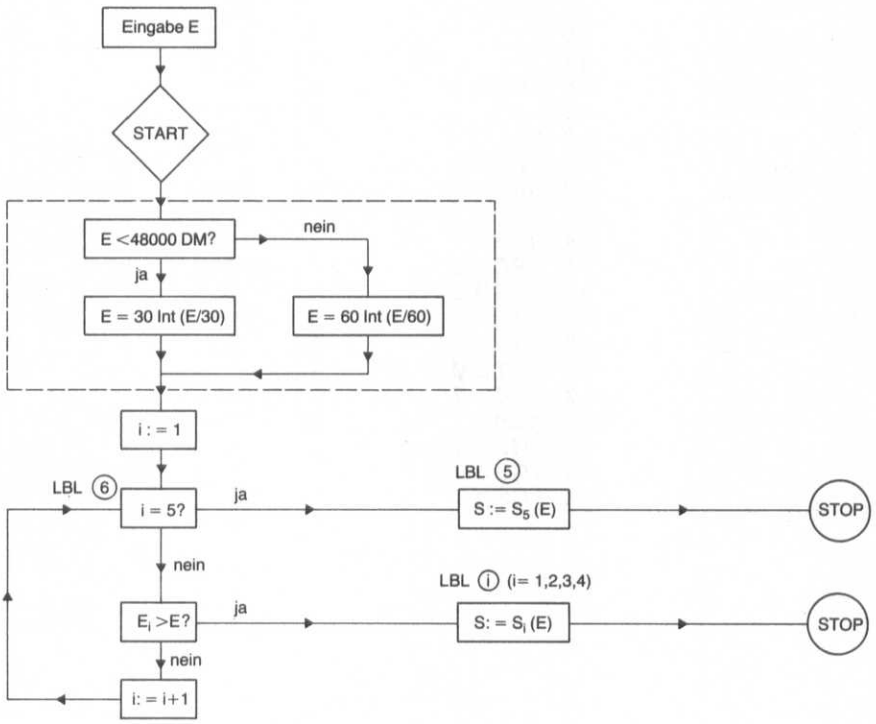
Literatur:

Einkommensteuergesetz (EStG) mit den neuesten im Bundesgesetzblatt Nr. 49 (1980) vom 21. August 1980 veröffentlichten Änderungen.

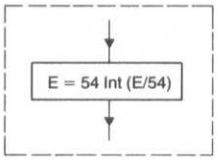
Tabelle 1: Konstanten der Steuerformel

	1979/80	1981
E_1	3691	4213
E_2	16001	18001
E_3	48000	60000
E_4	130000	130000
ε_1	16000	18000
ε_2	48000	60000
a_0	−812	−926
a_1	0.22	0.22
b_0	2708	3034
b_1	2200	2200
b_2	925	695
b_3	−154.42	−73.76
b_4	10.86	3.05
c_0	15298	20018
c_1	4800	5040
c_2	109.95	88.13
c_3	−6.07	−5.45
c_4	0.10	0.09
d_0	−13644	−14837
d_1	0.56	0.56

Flußdiagramm für die Berechnung der Steuerschuld



In dieser Form gilt das Flußdiagramm für 1980. Für 1981 braucht nur der eingerahmte Teil des Diagramms ersetzt zu werden durch:



(bitte umblättern)

Berechnung der Einkommensteuer für HP 29C

HP 29-C-Taschenrechnerprogramm zur Berechnung der Steuerschuld aus dem zu versteuernden Jahreseinkommen (Grund- oder Splittingtabelle)

Variante 1: Gültig für die Steuerjahre 1980 und 1981

```
Ø1 gLBL Ø
Ø2  RCL.3
Ø3  ÷
Ø4  STO.1
Ø5  RCL 3
Ø6 fx>y
Ø7  GTO 1
Ø8  RCL.2
Ø9  2 oder 1[1)]
1Ø  x
11  GTO 2
12 gLBL 1
13  RCL.2
14 gLBL 2
15  STO Ø
16  RCL.1
17  x⇌y
18  ÷
19 fINT
2Ø  RCL Ø
21  x
22  STO.1
23  1
24  STO Ø
25 gLBL 6
26  RCL Ø
27  5
28 fx=y
29  GTO 5
3Ø  RCL.1
31  RCL i
32 fx>y
33  GTO i
34 gISZ
35  GTO 6
36 gLBL 1
37  Ø
38  GTO 9
39 gLBL 2
4Ø  7
41  STO+Ø
42  RCL 8
43  GTO 8
44 gLBL 3
45  RCL 2
46  1
47  -
48  GSB 3
49  1
5Ø  5
51  STO Ø
52  RCL.Ø
53  GTO 7
54 gLBL 4
55  RCL 3
56  GSB 3
57  1
58  9
59  STO Ø
6Ø  RCL 7
61 gLBL 7
62  GSB 7
63  GSB 7
64  GSB 7
65 gLBL 8
66  GSB 7
67 fINT
68  RCL.3
69  x
7Ø gLBL 9
71 gRTN
72  GTO Ø
73 gLBL 5
74  RCL 6
75  GTO 8
76 gLBL 7
77  RCL.1
78  x
79  RCL.4
8Ø  x
81 fINT
82  RCL.4
83  ÷
84  RCL i
85  +
86 gISZ
87 gRTN
88 gLBL 3
89  STO-.1
9Ø  EEX
91  4
92  STO÷.1
93 gRTN
```

Speicherinhalte:

STO	1980	1981
0	—	—
1	3691	4213
2	16001	18001
3	48000	60000
4	130000	130000
5	−13644	−14837
6	.56	.56
7	.1	.09
8	.22	.22
9	−812	−926
.0	10.86	3.05
.1	—	—
.2	30	54
.3	1 oder	2[2)]
.4	1000	1000
.5	−154.42	−73.76
16	925	695
17	2200	2200
18	2708	3034
19	−6.07	−5.45
20	109.95	88.13
21	4800	5040
22	15298	20018
23 ⋮ 29	unbenützt	

Speicherregister ⓪ wird als Zwischenspeicher und als Adreßregister für die nur indirekt zugänglichen Speicherplätze ⑯ bis ㉒ benützt.
Speicherregister ① wird als Zwischenspeicher für Polynomberechnungen benötigt.
Die indirekt zugänglichen Speicherregister ㉓ bis ㉙ bleiben unbenützt.

[1)] 2 für 1980, 1 für 1981.
[2)] ③ = 1 für die Grundtabelle, 2 für die Splittingtabelle

Vorbereitungsschritte:

1) Programm eintasten.

2) Daten ① bis ⑤ (ohne ①) eingeben.

3) Eingabe der Daten ⑯ bis ㉒ (z.B. für 1981) durch die **Tastenfolge:**

1/6/STO 0/6/9/5/STO
i/gISZ/2/2/0/0/STO
i/gISZ/3/0/3/4/STO
i/gISZ/5/./4/5/CHS/STO
i/gISZ/8/8/./1/3/STO
i/gISZ/5/0/4/0/STO
i/gISZ/2/0/0/1/8/STO i

4) Zu versteuerndes Einkommen E eingeben, Programm starten mit GSB 0. Ab der 2. Eingabe E genügt R/S zum Programmstart.

Zu beachten:

1) Rechner nicht zwischendurch abschalten, da sonst die Daten 16 bis 22 verlorengehen und dann wieder neu eingetastet werden müssen.
2) Das Programm benutzt die Label 1, 2, 3 und 7 doppelt. Dies ist durchaus erlaubt, wenn nur sichergestellt ist, daß von der Sprunganweisung GTO die jeweils richtige Marke angesprungen wird. Bei Programmänderungen ist Vorsicht geboten!

Variante 2: Gültig für das Steuerjahr 1981

Die Programmzeilen 04 bis 20 von Variante 1 brauchen nur ersetzt zu werden durch die Zeilen

04 RCL.2
05 ÷
06 fINT
07 RCL.2

Die Zeilen 21 bis 93 von Variante 1 erhalten in Variante 2 die Nummern 08 bis 80.

Speicherinhalte und Vorbereitungsschritte wie bei Variante 1.

Tabelle 2: Einige Werte aus der Grund- und Splittingtabelle zur Programmkontrolle

zu verst. Einkommen	Steuerschuld Grundtabelle 1980	1981	Splittingtabelle 1980	1981
2 500	0	0	0	0
5 000	283	166	0	0
10 000	1 385	1 271	566	332
15 000	2 488	2 364	1 676	1 426
20 000	3 720	3 496	2 770	2 542
25 000	5 328	4 874	3 866	3 636
30 000	7 218	6 542	4 976	4 728
35 000	9 300	8 442	6 110	5 846
40 000	11 537	10 506	7 440	6 992
45 000	13 869	12 742	8 976	8 296
50 000	16 252	15 069	10 656	9 748
55 000	18 690	17 510	12 474	11 370
60 000	21 206	20 015	14 436	13 084
65 000	23 736	22 539	16 476	14 912
70 000	26 307	25 132	18 600	16 884
75 000	28 945	27 733	20 822	18 906
80 000	31 584	30 394	23 074	21 012
90 000	36 978	35 771	27 738	25 484
100 000	42 428	41 236	32 504	30 138
125 000	56 345	55 140	44 900	42 544
150 000	70 356	69 139	57 890	55 466
175 000	84 333	83 140	71 224	68 850
200 000	98 344	97 141	84 856	82 472
250 000	126 333	125 143	112 690	110 280
300 000	154 356	153 146	140 712	138 278
400 000	210 333	209 150	196 688	194 282
500 000	266 344	265 155	252 666	250 286

Heinz Dibold

Apple II und Apple II plus als Textsystem

Aufgabenstellung

Das hier veröffentlichte Programm wurde entwickelt, um den Mikrocomputer APPLE II als „Textsystem" mit 80 Zeichen je Zeile, deutschem Zeichensatz, Groß- und Kleinschrift und mit Steuerzeichen im Text benutzen zu können. Eine Forderung hierbei war, ohne Hardwareeingriffe und ohne zusätzliche „80-Zeichen-Karte" auszukommen.

Grundlegende Probleme und deren Lösung

Die Darstellung von Text am Bildschirm des APPLE II erfolgt durch 24 Zeilen zu je 40 Zeichen. Für die Textverarbeitung ist jedoch eine Zeilenlänge bis zu 80 Zeichen erforderlich. Das vorliegende Programm faßt daher je zwei Bildschirmzeilen zu einer Textzeile zusammen.

Die Tastatur des APPLE II erzeugt nur Großbuchstaben, es fehlen auch Tasten für spezielle Zeichen des deutschen Zeichensatzes. Diesem Umstand wurde durch Software-mäßige Umcodierung der Tastencodes Rechnung getragen. Die Taste ESC (vor dem Großbuchstaben) ersetzt die Funktion der Taste SHIFT (während des Buchstabens). Auch können über ESC alle 32 ASCII-Steuerzeichen (0—32) dezimal eingegeben werden.

Am Bildschirm des APPLE II können keine Kleinbuchstaben dargestellt werden. Das Programm TEXT 1.0 stellt daher Kleinbuchstaben in normaler Großschrift dar, während Großbuchstaben in inverser Schrift dargestellt werden. Invers werden auch sonst nichtsichtbare Steuerzeichen als entsprechende Zeichen mit um 32 höherem ASCII-Code dargestellt.

Programmbeschreibung

Das Programm „führt" den Benutzer mittels einer HEADLINE (oberste Zeile), in der die Kommandoebene und die zugehörigen Steuerzeichen angezeigt werden.

Die Textverarbeitung erfolgt bildschirmorientiert, d.h., bei der Texteingabe kann der am Bildschirm dargestellte Textausschnitt (11 Zeilen mit bis zu 80 Zeichen) durch den frei beweglichen Cursor gestaltet und korrigiert werden.

Dabei wird die Einhaltung der vorgebbaren „Textgrenzen" (Rand links, rechts, oben und unten) überwacht. Bei Annäherung an den rechten Rand erfolgt ein akkustischer Hinweis, bei Überschreitung des Randes wird automatisch getrennt, als Trennzeichen dient das letzte Leerzeichen oder der letzte Bindestrich. Das „Ende" einer Zeile (maximale Zeichenzahl erreicht) wird durch die Zeilennummer gekennzeichnet, die stets die Position der Zeile innerhalb der Seite angibt.

Nach Eingabe des Textes kann der Text am Bildschirm geprüft werden. Dazu wird ein Textausschnitt von elf Textzeilen dargestellt, dieser Ausschnitt kann frei über den gesamten Text verschoben werden. An jeder beliebigen Textstelle können Zeichen geändert, gelöscht oder hinzugefügt werden. Auch das Einfügen oder Löschen ganzer Abschnitte ist möglich.

Der so erstellte Text kann auf Diskette (oder Tonband) gespeichert werden.

Ein im Speicher befindlicher Text (eingegeben oder von Diskette oder Tonband geladen) kann an den Drucker ausgegeben werden. Die Ausgabe erfolgt entweder durch Vorgabe der Seitenanzahl mit der Möglichkeit der automatischen Seitennumerierung, oder durch wiederholte Ausgabe des gesamten Textes.

Beschreibung der Kommandos und zugehöriger Steuerzeichen

Kommando-Hauptebene „TEXT 1.0 . . ."

Sichten des vorhandenen Textes und Aufruf der einzelnen Kommandos. Die Kommandos sind:

„B" BEGIN
Cursor auf den Textbeginn setzen und Textausschnitt ausgeben.

„C" CHANGE
Ändern des Zeichens, auf das der Cursor zeigt.

„E" ENTER
Einfügen eines Zeichens, unmittelbar vor dem Cursor.

„I" INSERT
Kommando zum Einfügen, Ändern und Löschen vom Text. Der beim Aufruf des Kommandos am Bildschirm sichtbare Text, von der obersten linken Ecke bis unmittelbar vor dem Cursor wird aus dem Text im Textspeicher entfernt und steht am Bildschirm zur Änderung zur Verfügung. Wird der Cursor vor dem Abschluß des INSERT-Kommandos bezüglich dieses Textausschnitts zurückbewegt, so wird dieser Text verkürzt oder ganz gelöscht, wird er vorbewegt, so wird dieser Text um den neu hinzukommenden Text ergänzt und wieder in den Textspeicher eingefügt.

Eine ausführliche Beschreibung ist weiter unten zu finden.

„K" KILL
Das Zeichen unter dem Cursor wird aus dem Text entfernt.

„L" LOAD
Ein Text wird von Diskette in den Textspeicher geladen. Die mitaufgezeichneten Randgrenzen und der Zeilenabstand werden dabei übernommen.

„M" MARGIN
Die Werte für den Rand links, rechts, oben und unten werden angezeigt und können geändert werden. Der Zeilenabstand für das Drucken des Textes kann ebenfalls geändert werden. Der Zeilenabstand und die Druckerinitialisierung sind auf den Drucker EPSON MX-80 abgestimmt. Eine Anpassung an andere Drucker ist weiter unten beschrieben.

„N" NEW
Löscht den Textspeicher durch Neuinitialisierung der internen Textzeiger.

„P“ PRINT
Ausgabe des im Textspeicher vorhandenen Textes an den Drucker. Die Ausgabe erfolgt seitenweise, wobei anzugeben ist, wieviele Seiten ohne Unterbrechung mit fortlaufender Seitennumerierung auszugeben sind. Nach jedem dieser „Blöcke“ kann dann mit einer neuen Seitennummer weitergedruckt werden. Wird eine negative Seitennummer eingegeben, so unterbleibt die Ausgabe einer Seitennummer.

„R“ REPEATED PRINTING
Der gesamte Text wird ohne Seitennumerierung und ohne Unterbrechung eine einzugebende Anzahl oft an den Drucker ausgegeben.

„S“ SAVE
Speichert den im Textspeicher befindlichen Text mit den zugehörigen Randwerten und dem Zeilenabstand auf Diskette ab.

„Z“ ZAP
Setzt den Cursor an das Textende und löscht den Bildschirm.

„<“
Verschiebt den Textausschnitt ab der Cursor-Position um elf Zeilen zurück.

„>“
Verschiebt den Textausschnitt ab der Cursor-Position um elf Zeilen nach vorn.

„ “ SPACE
Gibt den Text ab der Cursor-Position am Bildschirm als neuen Textausschnitt aus.

„CR“ CARRIAGE RETURN
Setzt den Cursor auf den Beginn der nächsten Zeile und „scrollt“ den Textausschnitt gegebenenfalls.

„←“ BACKSPACE
Bewegt den Cursor ein Textzeichen zurück. Das letzte, nicht sichtbare Zeichen einer Zeile ist das Zeichen CR, falls die Zeile nicht durch einen Seitenvorschub (Inverses Komma) abgeschlossen ist.

„→“ NAK
Bewegt den Cursor um ein Textzeichen nach vorn.

Steuerzeichen des INSERT-Kommandos:

CR
Cursor auf den Beginn der nächsten Textzeile setzen. Wird dabei die letzte Bildschirmzeile erreicht, so erfolgt das Einfügen der obersten Textzeile in den Textspeicher und das „Scrollen“ des Bildschirms um Platz für eine neue Textzeile zu haben.

←
Bewegt den Cursor um eine Zeichenposition zurück.

→
Bewegt den Cursor eine Zeichenposition nach vorn. Wird dabei die unterste Bildschirmzeile erreicht, dann weiter wie bei CR.

CTRL/I
Bewegt den Cursor um eine Bildschirmzeile nach oben.

CTRL/J
Bewegt den Cursor um eine Bildschirmzeile nach unten. Wird dabei die unterste Zeile erreicht, dann weiter wie bei CR.

CTRL/D
Beendet das INSERT-Kommando. Der noch am Bildschirm sichtbare Text wird bis unmittelbar vor die Cursor-Position in den Textspeicher eingefügt. Alle Textzeilen vor dem Cursor werden von unnötigen Leerzeichen befreit und mit CR abgeschlossen. Enthält eine Zeile das Zeichen für Seitenvorschub (Inverses Komma), so wird die Zeile durch dieses Zeichen abgeschlossen.
Die Textzeile, in der der Cursor stand, wird nur bis zum Cursor berücksichtigt, nicht von Leerzeichen befreit und nicht durch CR abgeschlossen.

CTRL/L
Bei Eingabe des Zeichens für Seitenvorschub wird dieses am Bildschirm abgelegt, der Rest der Textzeile am Bildschirm gelöscht und die Zeilennumerierung am Bildschirm berichtigt.

ESC
Ersatz für die fehlende Groß-/Kleinschriftumschaltung der Tastatur. Großbuchstaben werden durch Drücken der ESCape-Taste vor dem Buchstaben erzeugt (statt SHIFT während des Buchstabens).
Mit ESC können auch Steuerzeichen (ASCII-Code < 32) erzeugt und damit im Text abgelegt werden. Hierzu ist der entsprechende Code als zweistellige Dezimalzahl nach dem Drücken der ESC-Taste einzugeben.

Jedes andere Zeichen
wird als Text aufgefaßt und am Bildschirm abgelegt, der Cursor wird nach rechts bewegt. Wird dabei der rechte Textrand überschritten, so werden die Zeichen bis zum letzten Leerzeichen oder Bindestrich zurück in die nächste Zeile verlagert. Ab sechs Zeichen vor dem rechten Rand wird akustisch auf den Rand hingewiesen, um durch das Setzen eines Bindestriches auch eine Worttrennung erfolgen zu lassen.

Darstellung des Textes

Der Text wird am Bildschirm wie folgt dargestellt:

- Großbuchstaben durch die entsprechenden Großbuchstaben in inverser, schwarzer Schrift auf hellem Hintergrund
- Kleinbuchstaben als entsprechende Großbuchstaben in normaler, weißer Schrift
- Zahlen und sonstige Textzeichen in normaler, weißer Schrift
- Steuerzeichen (ASCII-Code < 32) in inverser Darstellung des im Code um 32 höherliegenden entsprechenden Zeichens
- Sonderzeichen des deutschen Zeichensatzes:

ä durch „linke, eckige Klammer“, Eingabe durch CTRL/A
ö durch „Schrägstrich“, Eingabe durch CTRL/O
ü durch „rechte, eckige Klammer“, Eingabe durch CTRL/P
ß durch „Unterstreichungsstrich“, Eingabe durch CTRL/S

Speicherung des Textes

Im Speicher wird der Text im ASCII-Code in dem durch LOMEM reservierten Bereich gespeichert und mittels interner Textzeiger verwaltet. Gegenüber der Verwendung von Stringvariablen bringt dies eine Speicherplatzersparnis und vor allem eine erhöhte Sicherheit vor Datenverlust bei ungewolltem Programmaussprung mit sich.
Während der Editierphase des INSERT-Kommandos dient der Bildwiederholspeicher der Seite 1 als Arbeitsspeicher. Der Wiedereintritt in das Textprogramm ohne Zeigerinitialisierung erfolgt durch „GOTO 20 000“
Auf Diskette wird der Text als Binärdatei im ASCII-Code gespeichert. Soll ein Tonband als externer Speicher dienen, so sind die Kommandos „SAVE“ and „LOAD“ entsprechend zu ändern.

(bitte umblättern)

Apple II und Apple II+ als Textsystem

Programmiersprache

Das Programm wurde in Integer-Basic geschrieben, um es relativ leicht an geänderte Ansprüche anpassen und erweitern zu können. Obwohl die Verwendung von Integer-Basic gegenüber Applesoft-Basic schon Geschwindigkeitsvorteile bringt, wurden einige zeitkritische Aufgaben als Unterprogramme in Maschinensprache geschrieben.

Anpassung an das Mikrocomputersystem

Das vorliegende Programm ist lauffähig auf einem APPLE II oder APPLE II+ mit 48K-RAM, Integer-Basic in ROM oder auf einer Sprachkarte und einem Floppy-Disk Laufwerk.
Bei Systemen mit weniger als 48K frei verfügbaren RAMs, sind die Werte für den Textspeicher (von BB bis BE) in Zeile 30030 entsprechend zu ändern. Die Maschinenprogramme von derzeit $7533 bis $75ee incl. sind entsprechend mitzuverschieben.
Das Umschreiben des Programms in Applesoft-Basic ist möglich, bringt aber weitere Geschwindigkeitseinbußen bei der Ausgabe des Textes mit sich.

Kommentare zum Programmlisting

90 bis 95
Zeichen aus der „Bildschirm"-Codierung in den ASCII-Code wandeln.

100
Text der obersten Textzeile (2 Bildschirmzeilen) in den Textspeicher einfügen, Bildschirm „scrollen" und neue Textzeile mit Zeilennummer versehen.

140 bis 180
Text der obersten Textzeile ohne unnützige Leerzeichen in den Textspeicher einfügen und mit CR (Carriage Return) abschließen. Enthält die Zeile ein Seitenvorschubzeichen (FF), dann wird die Zeile bis zu diesem Zeichen ausgewertet und durch FF abgeschlossen. Bildschirm um 2 Bildschirmzeilen nach oben „scrollen".

200 bis 205
Text im Textspeicher ab TP um S Bytes nach „rechts" verschieben, TE neu berechnen.

250 bis 260
S Zeichen des Textes ab TP aus dem Textspeicher entfernen, TE neu berechnen.

300
Zeichen in C vom ASCII-Code in den „Bildschirmcode" wandeln und an den Bildschirm ausgeben

350
Zeiger H auf den Text im Textspeicher ab H+1 bis zum nächsten CR oder FF oder bis zum TE erhöhen

400 bis 410
Zeiger H auf den Text im Textspeicher ab H+1 bis zum nächsten CR oder FF oder bis zum TE erhöhen

400 bis 410
Zeiger H auf den Text im Textspeicher ab H−1 bis zum nächsten CR oder FF oder bis zum TB erniedrigen

520 bis 520
Zeiger PP auf TP setzen (für INSERT) und elf Zeilen Text ab TP am Bildschirm ausgeben

600
Bildschirm „scrollen" und neue Zeile aus dem Textspeicher am unteren Bildschirmrand ausgeben

610
Ausgabe einer Textzeile ab TP aus dem Textspeicher an den Bildschirm ab der momentanen Cursor-Position

650 bis 660
Wie 610, löscht jedoch den nicht benutzten „Rest" der Textzeile und ändert die Cursorposition nicht

700 bis 710
TP um elf Zeilen zurückbewegen
800 bis 810
TP um elf Zeilen vorbewegen

900
7-bit-ASCII-Zeichen von der Tastatur holen und in Kleinbuchstaben wandeln, nach ESC-Eingabe Buchstaben als Großbuchstaben lassen oder Steuerzeichen (ASCII-Code 0 bis 32) erzeugen

1100 bis 1190
Eine Seite Text an den Drucker ausgeben. Ist das erste Zeichen beim Aufruf ein FF und die Seitennummer nicht negativ, dann wird mittig SN als Seitennummer ausgegeben und erhöht. Der obere Rand wird durch Zeilenvorschübe, der linke Rand durch Leerzeichen erzeugt. Die Ausgabe endet mit Seitenvorschub durch das Zeichen FF (Form Feed)

1170 bis 1180
Gibt einen Zeilenvorschub mit dem errechneten Restabstand aus, der benötigt wird, um durch Seitenvorschub genau auf den nächsten Seitenbeginn zu treffen

2000 bis 2010
Stellt die Zeilennumer der Zeile, auf die der TP zeigt, fest und übergibt sie in Z

2100
Gibt in Abhängigkeit von LL und R1 zwei verschiedene Töne aus, Warnung bei Erreichen des rechten Randes

2110
Ton als Warnung bei Überschreitung des unteren Randes einer Textseite

2200 bis 2230
Initialisierung des Druckers, hier für EPSON MX-80 mit EPSON-Interface für den APLLE II. Die momentane Papierposition wird Seitenanfang, der Zeilenabstand und die errechnete Zeilenanzahl je Seite werden programmiert.

3000 bis 3020
Cursor auf den Beginn der nächsten Zeile setzen, dabei den Rest der alten Zeile löschen. Wird die unterste Bildschirmzeile erreicht, so wird der Bildschirm „gescrollt" und die neue Zeile ausgegeben. Der Zeiger PP für den Textausschnitt wird dabei um eine Zeile vorbewegt (für INSERT).

4000 bis 4010
Randwerte, Zeilenabstand und Textzeiger im RAM ablegen

4050 bis 4060
Randwerte, Zeilenabstand und Textzeiger aus dem RAM holen und Max. Zeilenzahl ZM sowie Zeilenabstandsrest ZABR aus ZAB neu berechnen.

5000 bis 5940
INSERT-Kommando

5000
Zeilengrenzen aus den Randwerten berechnen

5010
HEADLINE ausgeben, Rest der Seite, ab dem Cursor, löschen

5020
Zeilennummer der Cursor-Zeile feststellen, am Bildschirm sichtbaren Textausschnitt aus dem Textspeicher entfernen.

5025
Rest der Seite mit Zeilennumerierung versehen, die Zeilennummer zeigt die absolute Zeilengrenze an.

5030—5040
Steuerzeichen oder Textzeichen einlesen, Sprungverteiler, Prüfung des Textspeichers auf bevorstehenden Überlauf, wenn zu befürchten, dann INSERT-Kommando abschließen.

5100
Textzeichen am Bildschirm ausgeben, Cursor eins weiter

5110 bis 5120
Aktuelle Zeilenlänge feststellen und auf Einhaltung des rechten Randes und der absoluten Zeilengrenze prüfen, gegebenenfalls Warnton ausgeben oder Zeilentrennung

5200 bis 5210
„CR", Cursor auf nächsten Zeilenbeginn setzen, beim Erreichen des unteren Randes oberste Textzeile in den Textspeicher aufnehmen. Bildschirm „scrollen", Zeilennummer erhöhen und ausgeben. Auf Einhaltung des unteren Randes und der absoluten Seitengrenze achten, gegebenenfalls Warnton ausgeben oder Seitenvorschub einfügen.

5300 bis 5310
Zeichen FF für Seitenvorschub am Bildschirm ausgeben, Rest der Zeile löschen und zur nächsten Zeile gehen, falls erforderlich, oberste Textzeile in den Textspeicher einfügen und Bildschirm scrollen.

5400
„< —", Cursor eine Zeichenposition zurück

5500
„— >„ Cursor eine Zeichenposition nach vorn

5600
„CTRL/J", Cursor eine Bildschirmzeile nach unten, bei Erreichen des unteren Randes die oberste Textzeile in den Textspeicher einfügen und den Bildschirm "scrollen", Zeilennummer erhöhen und ausgeben.

5700
Cursor eine Bildschirmzeile höher setzen

5800 bis 5840
Automatische Zeilentrennung, die letzten Zeichen werden bis zum letzten Leerzeichen oder Bindestrich zurück in die nächste Zeile verlagert. Wenn das „Wort" zu lang oder die Zeile zu kurz ist, so unterbleibt die Trennung.

5890
Selbständiger Abschluß des INSERT-Kommandos bei Gefahr des Überlaufs des Textspeichers, „BUFFER FULL" melden.

5900 bis 5940
„CTRL/D", Abschluß des INSERT-Kommandos. Die Textzeilen „links" des Cursors werden in den Textspeicher eingefügt, die neuen Textzeiger werden im RAM hinterlegt.

6000 bis 6040
„CHANGE", das Zeichen, auf das der Cursor zeigt, wird geändert, bei der Eingabe von CR oder FF erfolgt Zeilenvorschub unter Löschung des Zeilenrestes, die jetzige, geänderte Zeile wird ausgegeben.

7000 bis 7020
„NEW", Textspeicher durch Neuinitialisierung der Textzeiger löschen.

7100 bis 7190
„MARGIN", neue Werte für die Randgrenzen (für INSERT) und den Zeilenabstand (für den Drucker) einlesen und im RAM hinterlegen, ZM und ZABR neu berechnen.

8000 bis 8030
„PRINT", Drucker-Interface und Drukker initialisieren, Text im Textspeicher in Blöcken mit eingebbarer Seitenanzahl und mit Seitennumerierung ausgeben, ist die Seitennummer negativ, so unterbleibt die Seitennumerierung.

9000 bis 9030
„REPEATED PRINTING", Wiederholtes Drucken des gesamten Textspeicher-Inhalts ohne Seitennumerierung.

10000 bis 10010
„SAVE", Inhalt des Textspeichers und aktuelle Parameter als Binärdatei auf Diskette speichern.

105000 bis 10510
„LOAD", Text von Diskette in den Textspeicher einlesen, die zugehörigen Parameter übernehmen und ZM sowie ZABR neu berechnen.

11000
„KILL", das Zeichen, auf dem der Cursor steht, aus dem Textspeicher entfernen, neuen Rest der Zeile am Bildschirm ausgeben.

11400 bis 11440
„ENTER", ein neues Zeichen unmittelbar vor dem Cursor in den Textspeicher einfügen. Wird CR oder FF eingegeben, so wird der neue Rest der Zeile am Bildschirm gelöscht, der Cursor auf die nächste Zeile gesetzt und diese neu ausgegeben.

20000 bis 20340
„TEXT 1.0", Sichten des Textes und Verzweigung zu den einzelnen Kommandos

20020
Text ab dem Cursor (TP) neu am Bildschirm ausgeben

20030 bis 20080
Kommando-Sprungverteiler

20100
„B", Cursor auf den Textanfang (TB) setzen, Text ausgeben

20120
„Z", Cursor auf das Textende (TE) setzen, Text ausgeben

20140 bis 20150
„< —", Cursor ein Textzeichen zurück

20200 bis 2010
„— >", Cursor durch Ausgabe des jetzigen Zeichens ein Zeichen nach vorn, wird dabei ein CR oder FF übergangen, so erfolgt Zeilenvorschub nach Löschen des Zeilenrestes am Bildschirm und Ausgabe der jetzt neuen Zeile (Kontrolle nach ENTER/KILL).

20250 bis 20270
CR, Rest der Textzeile ab der Cursorposition ausgeben bis CR oder FF gefunden wird, dann Rest der Zeile am Bildschirm löschen und zu Beginn der nächsten Zeile gehen. Auf diese Weise stimmt der Text "links" des Cursors stets mit dem tatsächlich im Textspeicher vorhandenen Text überein, dies ist für den Aufruf von INSERT erforderlich.

(bitte umblättern)

Apple II und Apple II+ als Textsystem

20300
„<“, Textausschnitt um elf Zeilen zurück

20320
„>“, Textausschnitt um elf Zeilen nach vorn

20340
„“, Text ab dem Cursor (TP) neu ausgeben

30000–30200
Programm-Initialisierung

30000
Speicherplatz für den Textspeicher (2048 bis 30000) und für die Maschinenprogramme (bis unter 32000) durch Setzen von LOMEM auf 30200 reservieren.

30005
Laden der Maschinenprogramme von Diskette in den RAM, die RAM-Anfangsadresse und die Aufzeichnungslänge sind im Dateinamen enthalten!

30010
Zeitkritische Variable zuerst anlegen

30020 bis 30025
Wertzuweisung der symbolischen Adreßnamen und ASCII-Codes

30030
Textspeichergrenzen (BB, BE) festlegen, Textanzeiger (TB, TP, TE) und Textausschnittszeiger (PP) initialisieren

30030 bis 30050
Symbolische Adressen der Maschinenprogramme definieren

30060
Randgrenzen und Drucker-Parameter (EPSON MX-80) festlegen

Programm-Listing in Integer-Basic
und
disassemblierter Ausdruck der Maschinenprogramme

```
>REM PROGRAMM-LISTING

>LIST
   10 GOTO 30000
   20 REM
   30 REM  TEXT 1.0, 09.04.1981
   40 REM  HEINZ DIBOLD
   50 REM  SOELLBACHDAMMWEG 15
   60 REM  8182 BAD WIESSEE
   70 REM
   90 IF C>0 AND C<30 THEN H=C+64
      : IF C>31 AND C<64 THEN H=C-
      32
   95 IF C>159 AND C<193 OR C=222
       THEN H=C-128: IF C>192 AND
      C<222 THEN H=C-96: IF C=223
       THEN H=126:C=H: RETURN
  100 IH= PEEK (CH)+1: GOSUB 140:
      Z=Z+1: VTAB 22+(R2+1)/40: TAB
      (R2+1) MOD 40+1: PRINT Z;: IF
      Z<10 THEN PRINT " ";: VTAB
      22: TAB IH: RETURN
  140 I=1152:S=0: FOR P=1 TO R2: IF
      I=1192 THEN I=1280:C= PEEK
      (I): IF C#160 THEN S=P: IF
      C=44 THEN 170:I=I+1: NEXT P:
      S=S+1: GOSUB 200
  150 IF S=1 THEN 160:I=1152: FOR
      TP=TP TO TP+S-2: IF I=1192 THEN
      I=1280:C= PEEK (I): GOSUB 90
      : POKE TP,C:I=I+1: NEXT TP
  160 POKE TP,CR:TP=TP+1: CALL SCROLL:
       CALL SCROLL: CALL UP: CALL
      UP: RETURN
  170 P=R2: NEXT P: GOSUB 200:I=1152
      : FOR TP=TP TO TP+S-1: IF I=
      1192 THEN I=1280:C= PEEK (I)
      : GOSUB 90: POKE TP,C:I=I+1
      : NEXT TP
  180 CALL SCROLL: CALL SCROLL: CALL
      UP: CALL UP: RETURN
  200 IF S<1 OR TE+S>BE THEN RETURN
      :TE=TE+S: IF TP+S=TE THEN RETURN
      : POKE 16,(TE-S) MOD 256: POKE
      17,(TE-S)/256
  205 POKE 18,TE MOD 256: POKE 19
      ,TE/256: POKE 20,(TP-1) MOD
      256: POKE 21,(TP-1)/256: CALL
      SHR: RETURN
  250 IF S<1 OR TE-S<BB THEN RETURN
      :TE=TE-S: IF TP=TE THEN RETURN
      : POKE 16,TP MOD 256: POKE
      17,TP/256
  260 POKE 18,(TP+S) MOD 256: POKE
      19,(TP+S)/256: POKE 20,TE MOD
      256: POKE 21,TE/256: CALL SHL:
       RETURN
  300 IF C=126 THEN H=223: IF C>96
       AND C<126 THEN H=C+96: IF
      C>31 AND C<65 OR C=94 THEN
      H=C+128: IF C<32 THEN H=C+32
      : IF C>64 AND C<94 THEN H=C-
      64: POKE CO,H: CALL OUT: RETURN
  350 POKE 4,H MOD 256: POKE 5,H/
      256: CALL ZEV:H= PEEK (5)*256
      + PEEK (4): RETURN
  400 POKE 4,H MOD 256: POKE 5,H/
      256: CALL ZER
  410 H= PEEK (5)*256+ PEEK (4): RETURN
  500 CALL HOME:PP=TP: IF TP>=TE THEN
      RETURN :I=0: FOR P=TP TO TE-
      1:C= PEEK (P): IF C=CR THEN
      520: GOSUB 300: IF C=FF THEN
      520
  510 NEXT P: VTAB 2: TAB 1: RETURN

  520 CALL CRLF: IF PEEK (CV) MOD
      2=0 THEN CALL CRLF:I=I+1: IF
      I>10 THEN P=TE: GOTO 510
  600 CALL CRLF: CALL CRLF: VTAB
      22: GOSUB 610: VTAB 22: TAB
      1: RETURN
  610 IF TP>=TE THEN RETURN : FOR
      P=TP TO TE-1:C= PEEK (P): IF
      C#CR THEN GOSUB 300: IF C=CR OR
      C=FF THEN P=TE: NEXT P: RETURN

  650 IV= PEEK (CV)+1:IH= PEEK (CH)
      +1: GOSUB 610: CALL CLEL: IF
      PEEK (CV) MOD 2=0 THEN 660:
       CALL CRLF: CALL CLEL
  660 VTAB IV: TAB IH: RETURN
  700 I=0:H=TP
  710 GOSUB 400:I=I+1: IF I<12 THEN
      710:TP=H+1: RETURN
  800 I=0:H=TP-1
  810 GOSUB 350:I=I+1: IF I<11 THEN
      810:TP=H+1: IF TP>TE THEN TP=
      TE: RETURN
  900 CALL IN:C= PEEK (CI): IF C=
      1 THEN C=91: IF C=15 THEN C=
      92: IF C=16 THEN C=93: IF C=
      19 THEN C=126: RETURN
 1000 GOSUB 900: IF C=ESC THEN 1010
      : IF C>64 AND C<94 THEN C=C+
      32: RETURN
 1010 GOSUB 900: IF C>64 AND C<94
       THEN RETURN :H=(C-48)*10: GOSUB
      900:C=H+C-48: IF C<0 OR C>31
       THEN C=63: RETURN
 1100 IF P>=PE THEN RETURN : IF PEEK
      (P)#FF THEN 1130:P=P+1: IF
      SN<0 OR RO<3 OR SM-RL-RR<8 THEN
      1115: FOR J=1 TO RO-2: PRINT
      : NEXT J
 1110 FOR J=1 TO RL+(SM-RL-RR)/2-
      3: PRINT " ";: NEXT J: PRINT
      "- ";SN;" -": PRINT :SN=SN+
      1: GOTO 1120
 1115 IF RO=0 THEN 1120: FOR J=1 TO
      RO: PRINT : NEXT J
 1120 IF RL=0 THEN 1130: FOR J=1 TO
      RL: PRINT " ";: NEXT J
 1130 IF P>=PE THEN RETURN : FOR
      I=P TO PE-1:C= PEEK (I): IF
      C=FF THEN 1160: POKE CO,C: CALL
      OUT: IF C=CR THEN 1150
 1140 NEXT I:P=I: RETURN
 1150 IF RL=0 THEN 1140: FOR J=1 TO
      RL: POKE CO,BL: CALL OUT: NEXT
      J: GOTO 1140
 1160 P=I:I=PE: NEXT I
 1170 IF ZABR=0 THEN 1190: POKE CO,
      ESC: CALL OUT: POKE CO,65: CALL
      OUT: POKE CO,ZABR: CALL OUT:
       POKE CO,ESC: CALL OUT: POKE
      CO,50: CALL OUT
 1180 POKE CO,CR: CALL OUT: POKE
      CO,ESC: CALL OUT: POKE CO,65
      : CALL OUT: POKE CO,ZAB: CALL
```

```
     OUT: POKE CO,ESC: CALL OUT:
      POKE CO,50: CALL OUT
1190 POKE CO,FF: CALL OUT: RETURN
2000 Z=-1:H=TP
2010 GOSUB 400:Z=Z+1:C= PEEK (6)
     : IF C#0 AND C#FF THEN 2010
     : IF C=FF THEN Z=Z+RO: RETURN
2100 POKE TONL,2: POKE TONF,175:
      IF LL=R1 THEN POKE TONF,50
     : CALL TON: RETURN
2110 POKE TONL,8: POKE TONF,80: CALL
     TON: RETURN
2200 PRINT I$;"80N";: POKE CO,ESC:
      CALL OUT: POKE CO,65: CALL
     OUT: POKE CO,ZAB: CALL OUT:
      POKE CO,ESC: CALL OUT: POKE
     CO,50: CALL OUT
2210 POKE CO,ESC: CALL OUT: POKE
     CO,67: CALL OUT: POKE CO,ZM:
      CALL OUT
2230 RETURN
3000 CALL CLEL: CALL CRLF: IF PEEK
     (CV) MOD 2#0 THEN 3020: CALL
     CLEL: CALL CRLF
3020 IF PEEK (CV)<23 THEN RETURN
     : GOSUB 600:H=PP-1: GOSUB 350
     :PP=H+1: RETURN
4000 POKE 2048,SM: POKE 2049,ZAB:
      POKE 2050,RL: POKE 2051,RR:
      POKE 2052,RU: POKE 2053,RO
4010 POKE 2054,TB MOD 256: POKE
     2055,TB/256: POKE 2056,TP MOD
     256: POKE 2057,TP/256: POKE
     2058,TE MOD 256: POKE 2059,
     TE/256: RETURN
4050 SM= PEEK (2048):ZAB= PEEK (
     2049):RL= PEEK (2050):RR= PEEK
     (2051):RU= PEEK (2052):RO= PEEK
     (2053)
4055 ZM=ZAG/ZAB:ZABR=ZAG-ZM*ZAB:
      IF ZABR#0 THEN ZM=ZM+1
4060 TB= PEEK (2054)+ PEEK (2055
     )*256:TP= PEEK (2056)+ PEEK
     (2057)*256:TE= PEEK (2058)+
      PEEK (2059)*256: RETURN
5000 IV= PEEK (CV)+1:IH= PEEK (CH)
     +1:R1=SM-RL-RR:R2=SM-RL:RO=
     R1-6
5010 POKE WT,0: VTAB 1: TAB 1: PRINT
     "INSERT  CR <- -> CTRL/I,J,L ESC
      CTRL/D  ": POKE WT,1: VTAB
     IV: TAB IH: CALL CLEP
5020 GOSUB 2000:S=TP-PP:TP=PP: GOSUB
     250
5025 FOR I=IV+IV MOD 2 TO 22 STEP
     2: VTAB I+(R2+1)/40: TAB (R2+
     1) MOD 40+1:Z=Z+1: PRINT Z;
     : IF Z<10 THEN PRINT " ";: NEXT
     I: VTAB IV: TAB IH
5030 IF PEEK (CV)*40>=BE-TE THEN
     5890: GOSUB 1000
5040 IF C=CR THEN 5200: IF C=BS THEN
     5400: IF C=NAK THEN 5500: IF
     C=FF THEN 5300: IF C=EOT THEN
     5900: IF C=LF THEN 5600: IF
     C=HT THEN 5700
5100 GOSUB 300
5110 V= PEEK (CV):LL=40*((V+1) MOD
     2)+ PEEK (CH): IF LL>R2 THEN
     5200: IF LL>R1 THEN 5800
5120 IF LL>RO THEN GOSUB 2100: GOTO
     5030
5200 CALL CRLF:V= PEEK (CV): IF
     V MOD 2=0 THEN CALL CRLF
5210 IF PEEK (CV)>22 THEN GOSUB
     100: IF Z>=ZM THEN 5300: IF
     Z>=ZM-RU THEN GOSUB 2110: GOTO
     5030
5300 POKE CO,FF1: CALL OUT: CALL
     CLEL: CALL CRLF:V= PEEK (CV)
     : IF V MOD 2#0 THEN 5310: CALL
     CLEL: CALL CRLF
5310 Z=RO: IF V>21 THEN GOSUB 140
     :IV= PEEK (CV)+1:IH= PEEK (
     CH)+1: GOTO 5025
5400 CALL LEFT:V= PEEK (CV):H=40
     *((V+1) MOD 2)+ PEEK (CH): IF
     H<=R1 THEN 5030: VTAB V+R1/
     40: TAB R1 MOD 40: GOTO 5030
5500 CALL RIGHT: GOTO 5110
5600 CALL DOWN: IF PEEK (CV)>22 THEN
     GOSUB 100: GOTO 5030
5700 CALL UP: GOTO 5030
5800 IF R1<20 THEN 5120: CALL LEFT:
     P= PEEK (40)+ PEEK (41)*256
     + PEEK (CH):V= PEEK (CV): VTAB
     V+V MOD 2:BAS0= PEEK (40)+ PEEK
     (41)*256: CALL CRLF
5810 BAS1= PEEK (40)+ PEEK (41)*
     256: CALL CRLF:I=P: IF PEEK
     (I)=BL1 THEN 5210
5820 FOR J=0 TO 18: IF I=BAS1-1 THEN
     I=BAS0+39:H= PEEK (I): IF H=
     BL1 OR H=173 THEN 5830:I=I-
     1: NEXT J:I=P-1:H=1: GOTO 5840

5830 H=J:J=18: NEXT J
5840 FOR H=H TO 1 STEP -1:I=I+1:
      IF I=BAS0+40 THEN I=BAS1: POKE
     CO, PEEK (I): CALL OUT: POKE
     I,BL1: NEXT H: GOTO 5210
5890 IV= PEEK (CV)-1:IH= PEEK (CH)
     : POKE WT,0: VTAB 1: TAB 1:
      PRINT "BUFFER FULL !!!": POKE
     WT,1: GOTO 5910
5900 IV= PEEK (CV)-1:IH= PEEK (CH)

5910 IF IV<2 THEN 5920: FOR J=1 TO
     IV/2: GOSUB 140: NEXT J
5920 S=IH+IV MOD 2*40: IF S>R2 THEN
     S=R2
5930 IF S=0 THEN 5940: GOSUB 200
     :I=1152: FOR TP=TP TO TP+S-
     1: IF I=1192 THEN I=1280:C=
      PEEK (I): GOSUB 90: POKE TP,
     C:I=I+1: NEXT TP
5940 GOSUB 4000: GOTO 20000
6000 IV= PEEK (CV)+1:IH= PEEK (CH)
     +1: POKE WT,0: VTAB 1: TAB
     1: PRINT "CHANGE A SINGLE CHARAC
     TER                  ": POKE WT,
     1: VTAB IV: TAB IH
6020 GOSUB 1000: POKE TP,C:TP=TP+
     1: IF C=CR THEN 6040: GOSUB
     300: IF C#FF THEN 20010
6040 GOSUB 3000: GOSUB 650: GOTO
     20010
7000 POKE WT,0: CALL HOME: PRINT
     "NEW ,CLEAR TEXTBUFFER": POKE
     WT,1
7010 INPUT "ALTEN TEXT VERGESSEN?"
     ,K$: IF K$#"J" AND K$#"JA" THEN
     20000
7020 TB=BB:TP=TB:TE=TB: GOSUB 4000
     : GOTO 20000
7100 POKE WT,0: CALL HOME: PRINT
     "MARGIN, SET NEW VARIABLES"
     : POKE WT,1
7105 PRINT "ZEILENABSTAND ";ZAB,
     : INPUT H: IF H<9 OR H>85 THEN
     7105:ZAB=H:ZM=ZAG/ZAB:ZABR=
     ZAG-ZM*ZAB: IF ZABR#0 THEN
     ZM=ZM+1
7110 PRINT "LINKER  RAND  ";RL,:
      INPUT H: IF H<0 OR H>=SM THEN
     7110:RL=H
7120 PRINT "RECHTER RAND  ";RR,:
      INPUT H: IF H<0 OR H>=SM-RL
     THEN 7120:RR=H
7130 PRINT "OBERER  RAND  ";RO,:
      INPUT H: IF H<0 OR H>=ZM THEN
     7130:RO=H
7140 PRINT "UNTERER RAND  ";RU,:
      INPUT H: IF H<0 OR H>=ZM-RO
     THEN 7140:RU=H
7190 GOSUB 4000: GOTO 20000
8000 POKE WT,0: CALL HOME: PRINT
     "PRINT": PRINT : POKE WT,1:
      INPUT "WIEVIELE SEITEN",PZ:
      IF PZ<1 THEN 20000:P=TB:PE=
     TE: INPUT "SEITENNUMMER ",SN
8010 PR#1: GOSUB 2200
8020 GOSUB 1100:PZ=PZ-1: IF PZ>0
      AND P<PE THEN 8020: PR#0
8030 IF P=PE THEN 20000: INPUT "WIEVI
     ELE SEITEN",PZ: IF PZ<1 THEN
     20000: INPUT "SEITENNUMMER "
     ,SN: GOTO 8010
9000 POKE WT,0: CALL HOME: PRINT
     "REPEATED PRINTING": PRINT
     : INPUT "WIE OFT ZU DRUCKEN"
     ,PZ: IF PZ<1 THEN 20000
9010 PR#1: GOSUB 2200
9020 P=TB:PE=TE:SN=-1
9030 GOSUB 1100: IF P<PE THEN 9030
     :PZ=PZ-1: IF PZ>0 THEN 9020
     : PR#0: GOTO 20000
10000 POKE WT,0: CALL HOME: PRINT
     "SAVE TEXT ON DISK": POKE WT,
     1: PRINT D$;"CATALOG": INPUT
     "TEXTNAME?",K$: IF K$="" THEN
     20000: GOSUB 4000
10010 PRINT : PRINT D$;"BSAVE ";K$
     ;",A2048,L";TE-2048: PRINT
     : PRINT D$;"VERIFY ";K$: GOTO
     20000
10500 POKE WT,0: CALL HOME: PRINT
     "LOAD TEXT FROM DISK": IF TE=
     TB THEN 10510: INPUT "ALTEN TEXT
      VERGESSEN?",K$: IF K$#"J" AND
     K$#"JA" THEN 20000
10510 POKE WT,1: PRINT D$;"CATALOG"
     : INPUT "TEXTNAME?",K$: IF
     K$="" THEN 20000: PRINT : PRINT
     D$;"BLOAD ";K$: GOSUB 4050:
      GOTO 20000
11000 S=1: GOSUB 250: GOSUB 650: GOTO
     20030
11400 IF TE>=BE THEN 20030
11420 IV= PEEK (CV)+1:IH= PEEK (CH)
     +1: POKE WT,0: VTAB 1: TAB
     1: PRINT "ENTER A SINGLE CHARACT
     ER                   ": POKE WT,
     1: VTAB IV: TAB IH
11430 GOSUB 1000:S=1: GOSUB 200: POKE
     TP,C:TP=TP+1: IF C=CR THEN
     11440: GOSUB 300: IF C=FF THEN
     11440: GOSUB 650: GOTO 20010

11440 GOSUB 3000: GOSUB 650: GOTO
     20010
20000 CALL HOME: VTAB 2
20010 IV= PEEK (CV)+1:IH= PEEK (CH)
     +1: POKE WT,0: VTAB 1: TAB
     1: PRINT "TEXT 1.0 B Z <-> CR BL
      E I K L M N P R S"
20015 POKE WT,1: VTAB IV: TAB IH:
      GOTO 20030
20020 GOSUB 500
20030 IF TP<TB THEN TP=TB: IF TP>
     TE THEN TP=TE: CALL IN:C= PEEK
     (CI): IF C=66 THEN 20100: IF
     C=90 THEN 20120: IF C=BS THEN
     20140
```

Apple II und Apple II+ als Textsystem

```
20040 IF C=60 THEN 20300: IF C=73
      THEN 5000: IF C=80 THEN 8000
      : IF C=82 THEN 9000: IF C=83
      THEN 10000: IF C=76 THEN 10500
20050 IF C=32 THEN 20340: IF C=78
      THEN 7000: IF C=77 THEN 7100
      : IF C=69 THEN 11400
20070 IF TP=TE THEN 20030: IF C=NAK
      THEN 20200: IF C=CR
      THEN 20250: IF C=
      62 THEN 20320: IF C=67 THEN
      6000: IF C=75 THEN 11000
20080 GOTO 20030
20100 TP=TB: GOTO 20020
20120 TP=TE: GOTO 20020
20140 IF TP=TB OR PEEK (CV)=1 AND
      PEEK (CH)=0 THEN 20030: CALL
      LEFT:TP=TP-1:C= PEEK (TP): IF
      C#FF AND C#CR THEN 20030
20150 H=TP: GOSUB 400:LL=TP-H-1: TAB
      (LL MOD 40)+1: IF LL<40 THEN
      CALL UP: GOTO 20030
20200 C= PEEK (TP):TP=TP+1: IF C=
      CR THEN 20210: GOSUB 300: IF
      C#FF THEN 20030
20210 GOSUB 3000: GOSUB 650: GOTO
      20030
20250 FOR P=TP TO TE-1:C= PEEK (P)
      : IF C=CR THEN 20260: GOSUB
      300: IF C=FF THEN 20260: NEXT
      P:TP=TE: GOTO 20270
20260 TP=P+1:P=TE: NEXT P
20270 GOSUB 3000: GOTO 20030
20300 IF TP=TB THEN 20030: GOSUB
      700: GOTO 20020
20320 GOSUB 800: GOTO 20020
20340 GOTO 20020
30000 POKE 74,30200 MOD 256: POKE
      75,30200/256: POKE 204,30200
      MOD 256: POKE 205,30200/256
      : PRINT "LOMEM AUF 30200"
30003 DIM D$(1): DIM I$(1):D$="":
      I$=": REM   CTRL/D, CTRL/I
30005 PRINT "MASCHINENPROGRAMME LADEN"
      : PRINT D$;"BLOAD BTEXT1.OA$7533
      L$BC"
30010 C=0:H=0:I=0:P=0:H=0:PA=0:PE=
      0:S=0:Z=0
30020 BL1=160:FF1=44:BL=32:CR=13:
      FF=12:BS=8:NAK=21:ESC=27:LF=
      10:HT=9:EOT=4:SO=14:BELL=7
30025 WT=34:CH=36:CV=37:PFL=50:INV=
      63:REG=255
30030 BB=2060:BE=30000:TB=BB:TP=TB:
      TE=TB:PP=TP:CI=2:CO=3
30040 LEFT=-1008:RIGHT=-1036:UP=-
      998:DOWN=-922:IN=30003:OUT=
      30011:SCROLL=-912:CLEP=-958
      :HOME=-936:CRLF=-926:LS=-16336

30050 CLEL=-868:TON=30085:TONF=0:
      TONL=1:SHL=30017:SHR=30048:
      ZEV=30127:ZER=30157:PR=4:CS=
      6: DIM K$(16)
30060 RL=15:RR=8:RO=6:RU=6:SM=79:
      ZAB=15:ZABR=12:ZM=53:ZAG=792

30150 GOTO 20000
30200 END
```

```
>REM HEX-DUMP DER MASCHINENPROGRAMME

>CALL-151

*7533.75EE

7533- 20 0C FD 29 7F
7538- 85 02 60 A5 03 20 ED FD
7540- 60 A0 00 B1 12 91 10 E6
7548- 10 D0 02 E6 11 E6 12 D0
7550- 02 E6 13 A5 10 C5 14 D0
7558- EA A5 11 C5 15 D0 E4 60
7560- A2 FF A0 00 B1 10 91 12
7568- C6 10 E4 10 D0 02 C6 11
7570- C6 12 E4 12 D0 02 C6 13
7578- A5 10 C5 14 D0 E6 A5 11
7580- C5 15 D0 E0 60 AD 30 C0
7588- 88 D0 04 C6 01 F0 07 CA
7590- D0 F6 A6 00 D0 EF 60 38
7598- A5 04 ED 06 08 A5 05 ED
75A0- 07 08 60 38 A5 04 ED 0A
75A8- 08 A5 05 ED 0B 08 60 A0
75B0- 00 F0 10 E6 04 D0 02 E6
75B8- 05 B1 04 C9 0D F0 0B C9
75C0- 0C F0 07 20 A3 75 90 EB
75C8- A9 00 85 06 60 A2 FF A0
75D0- 00 F0 12 C6 04 E4 04 D0
75D8- 02 C6 05 B1 04 C9 0D F0
75E0- 0B C9 0C F0 07 20 97 75
75E8- B0 E9 A9 00 85 06 60
*
```

DATA BECKER DATA BECKER DATA BECKER DATA BECKER DATA BECKER DATA

Sie interessieren sich für Kleincomputer?

Wir zeigen in einer ständigen Großausstellung auf über 450 qm Kleincomputer für alle Anwendungsbereiche (für Beruf, Hobby und Wissenschaft), vielseitige Peripheriegeräte, eine umfassende Zubehörauswahl, eine riesige Softwarepalette von der Systemroutine bis zur ausgereiften kommerziellen Lösung für zahlreiche Branchen, sowie Europas größte Auswahl an EDV-Literatur mit ständig neuen Titeln und Publikationen. Wir führen mit Ihnen ein Expertengespräch oder machen Sie mit der EDV vertraut. Wir präsentieren Ihnen unsere Leasing und Finanzierungsmöglichkeiten und unterbreiten Ihnen ein leistungsfähiges Angebot für Ihre bisherige Anlage. Oder kurz gesagt:

Das komplette EDV-Angebot

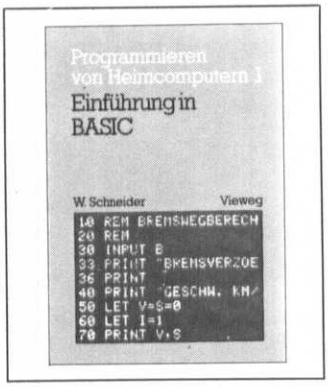
DATA BECKER

DATA BECKER GMBH im Hause Auto Becker
Merowinger Straße 30 · 4000 Düsseldorf
Telefon (0211) 312085 · Telex 08582874

DATA BECKER DATA BECKER DATA BECKER DATA BECKER DATA BECKER DATA

BECKER DATA BECKER DATA BECKER

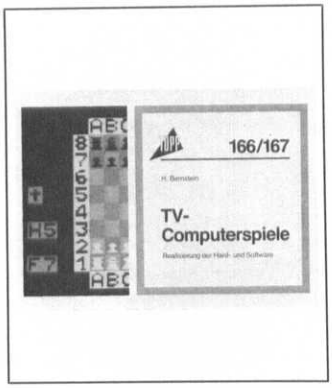
BECKER DATA BECKER DATA BECKER



Programmieren von Heimcomputern 1 Einführung in BASIC

Schneider, Wolfgang, 148 Seiten, 24 DM

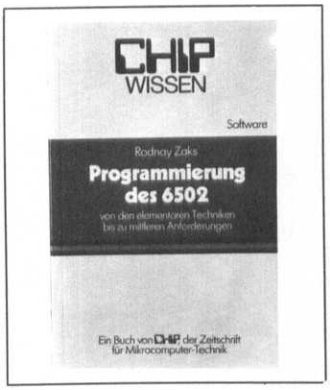
In den einzelnen Kapiteln wird besonderer Wert auf einprägsame Merksätze und lehrreiche Beispiele gelegt. Am Ende eines jeden Kapitels ist das Wichtigste zum Nachschlagen und zum Wiederholen konzentriert wiedergegeben. Vollständig programmierte Beispiele zeigen darüber hinaus, wie man eigene Programme entwickelt.



TV-Computerspiele

Von Herbert Bernstein, 208 Seiten, 21 DM.

Detaillierte Beschreibung der Hard- und Software für den Bau eines freiprogrammierbaren Fernsehcomputers. Es sind dazu nur wenige Bausteine erforderlich, mit Platinenentwürfen, Stücklisten, Abgleichvorschriften usw. Dieses System läßt sich beliebig erweitern und ist als intelligentes Fernsehspiel oder Heimcomputer einzusetzen. Einfache Programmierung.



Programmierung des 6502

von Rodnay Zaks, 355 Seiten, 44 DM

Die Programmierung des 6502 wird hier von den elementaren Techniken bis zu den mittleren Anforderungen beschrieben. Da die meisten Heimcomputer den Mikroprozessor 6502 als CPU besitzen, ist dieses Buch fast Pflichtlektüre für jeden Anwender, der ernsthaft Programme schreiben muß oder will.

Mikroprozessor-Interface-Techniken

Von Lesea und Zaks, 440 Seiten, 44 DM

Seit 1976 zu den Mikroprozessoren die zugehörigen Interfaceschaltkreise in LSI-Technik eingeführt wurden, ist die Schnittstellentechnik von Interfaces keine besondere Kunst mehr. Es geht jetzt nur noch um die Anwendung einiger Techniken, ja in manchen Fällen nur noch um einige Bauteile. Dieses Buch zeigt alles, was für die Erstellung eines vollständigen Systems benötigt wird.

Programmieren mit PASCAL

R. Baumann, 260 Seiten, 20 DM

Die ideale Einführung in die Kunst des Programmierens. Die Sprachkomponenten von PASCAL werden hier nicht systematisch abgehandelt, sondern beim Lösen konkreter Aufgaben nach und nach erarbeitet. Zahlreiche Übungsaufgaben (teils mit Lösungen) dienen der Festigung des Gelernten und laden zum Weiterdenken ein. Es werden hier keine Vorkenntnisse vorausgesetzt; an Mathematikkenntnissen genügen die der Mittelstufe. Selbststudium für den Hobby-Programmierer.

Mein Computer: 2^{10} Fragen und Antworten

Rich Diffay, 240 Seiten, 36 DM

Ein ausführlicher Überblick über die Computer- und Mikrocomputertechnik.

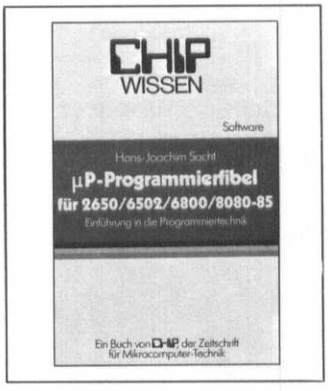
Der in Form eines Dialogs angelegte Text ist in Tage unterteilt. In Tage, an denen sich der Computerfachmann mit dem Anfänger unterhält. So wird der Leser systematisch auf das Ziel gelenkt, ein sicheres Gefühl für die Zusammenhänge im Bereich der Hobby- und Arbeitsplatzcomputer zu gewinnen. Dem Leser wird vom Anfang bis zum Ende die Möglichkeit gegeben, sich mit Problemen auseinanderzusetzen.

Was der Mikrocomputer alles kann

Willis, Pohl, 360 Seiten, 33 DM

Für alle Einsteiger und Aufsteiger. Ohne unnötigen technischen Ballast, leicht faßlich und schnell verständlich, unterhaltsam und umfassend. Genau das richtige Buch, um unkompliziert eine komplizierte Welt verstehen zu lernen.

Einführung, CPU und Drumherum. Woraus Speicher bestehen, die Daten-Supermärkte, sprechen mit dem Rechner, Interfaces, Grundzüge des Programmierens, Marktübersicht, Computerspiele, kommerzielle Anwendung, Künstlerisches und Augen auf beim Kauf.



µP-Programmierfibel für 2650/6502/6800/8080–8085

H. J. Sacht, 366 Seiten, 38 DM

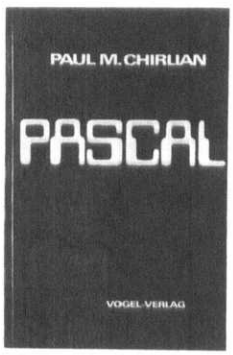
Eine leicht faßliche Einführung in die Programmiertechnik für Mikroprozessoren. Für alle, die noch nichts oder nur wenig vom Programmieren wissen.

Grundlagen für die Programmierarbeit, etwas Binär-Mathematik, Aufbau und Wirkung der Befehle für Mikroprozessoren, die Befehlssätze der µP 2650, 6502, 6800 und 8080 (85), Beispiele für Mikroprozessor-Programme (Lernprogramme) und Hinweise für die praktische Programmierarbeit.

Programming the Z80

Zaks, Rodnay, 624 Seiten, 50 DM (in Englisch)

Detaillierte Darstellung des Z80 mit vielen Bildern und verständlichen Erklärungen für einen weiten Leserkreis. Ein „Muß" für Z80-Anwender. Inhaber Z80-Hardware-Organisation, Programmiertechnik, Befehle, Adressierungsarten und -techniken, E/A-Techniken, Maschinenprogramme, Datenstruktur und Programmentwicklung.



PASCAL

Chirlian, Paul M., 212 Seiten, 30 DM (in Englisch)

Anfänger, die von Computern und Programmierung noch nichts wissen, werden auf einfache Weise mit PASCAL so vertraut gemacht, daß sie sofort Programme schreiben lernen, unabhängig vom Computertyp. Das Buch gibt einen guten Überblick über Struktur und Besonderheiten von PASCAL.

The Best of Micro

Von Micro Ink, 224 Seiten, 32 DM

Dieses Handbuch informiert den Leser über die Computer AIM 65, Apple II, Kim-1, CBM (PET), Superboard und SYM-1. Der CHIP-Leser kennt sofort hieraus, um welchen Mikroprozessor es sich handelt, nämlich um den 6502. Etwa auf 55 Seiten sind Programme für AIM/SYM/KIM, 49 Seiten für den Apple und 41 Seiten für den PET vorhanden. Auf 69 Seiten findet man noch weitere interessante Informationen.

Your First Computer

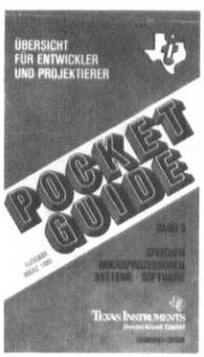
von Rodnay Zaks, 258 Seiten, 28 DM

Dieses englische Buch ist relativ einfach zu lesen, auch wenn man die Sprache nur ungenügend beherrscht. Es werden die Möglichkeiten für die Büro- und Personalcomputer besprochen, um später die Auswahl zu erleichtern. Neben der Beschreibung der einzelnen Baugruppen, wie CPU, Bildschirme, Speichermedien, Drucker usw. findet man zahlreiche Orientierungspunkte für die Zusammenstellung einer optimalen Anlage.

Heim-Schachcomputer

Einführung, Marktübersicht und Vergleichstest. Von Björn Schwarz, 182 Seiten, 19,80 DM

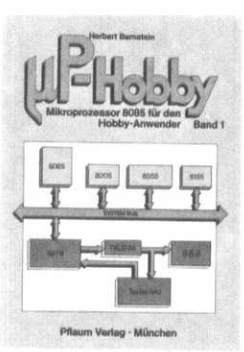
Wer sich mit Schachcomputern beschäftigt, wird sich bald dieses Buch kaufen. Neben der Entwicklung, die bereits seit 1769 die Schachspieler begeistern, wird die Funktionsweise der elektronischen Schachcomputer behandelt. Mit dem Kapitel „Worauf ist beim Kauf eines Heim-Schachcomputers zu achten?" erspart man viel Geld bei der Anschaffung.



Pocket Guide

von Texas Instruments, 517 Seiten, 22,50 DM

Der Band 3 der Pocket Guide-Serie behandelt die Speicher, Mikroprozessoren, die TI-Systeme und die Software. Diese Übersicht ist für den Entwickler und Projektierer ebenso geeignet, wie für den Hobby-Anwender.



µP-Hobby

Mikroprozessor 8085 für den Hobby-Anwender Band 1. Von Herbert Bernstein, 221 Seiten, 36 DM

In Verbindung mit der Firma Intel entstand ein Mikroprozessorbuch für den Hobbyisten. Dieser findet damit einen leichten Einstieg in die Computerei. Neben einer Hardware-Beschreibung sind auch zahlreiche Software-Probleme beschrieben. Hierzu gehört auch ein Kassetten-, CRT-Interface, TTY-Schnittstellen und der Aufbau eines Minimalsystems.

Mikrocomputer-Grundwissen

Allgemeinverständliche Einführung in die Mikrocomputer-Technik
Osborne, Adam, 300 Seiten, 36 DM

Deutsche Erstauflage eines Osborne-Bestsellers für Elektronikneulinge. Ohne großen Zeit- und Kostenaufwand kann im Selbststudium das Grundwissen erarbeitet werden. Nach sechs Lernschritten beherrscht der Leser die Materie und kann mitreden, wenn es um den Mikrocomputer geht.

Mikroprozessorlehre

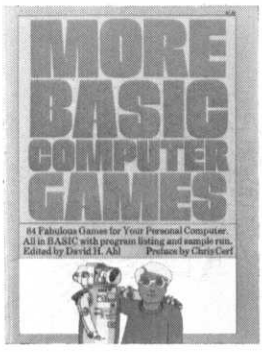
Starke, 282 Seiten, 38 DM

Dieses Buch behandelt den Mikroprozessor 8080 mit seinen Funktionen, Befehlslisten und seiner Programmierung. Neben einer ausführlichen Erklärung findet der Leser auch Testfragen in jedem Kapitel, die sein Wissen festigen. Besonders umfangreich ist die Programmierung des 8080, die Adressierung, Programmverzweigungen und Programmschleifen. Ideal für die berufliche Weiterbildung.

Mein Computer, Band 2: Software

Rich Didday, 230 Seiten, 38 DM

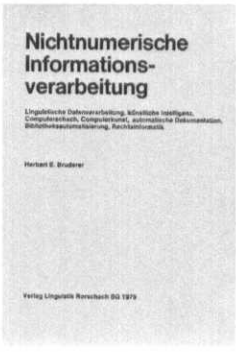
Dies ist der zweite Band von Didday, mit dem zwei Zwecke erfüllt werden sollen. Zuerst sollten sie dem Leser ein Gefühl für die Zusammenhänge in dem Bereich der Hobby- und Arbeitsplatzcomputer geben, so daß man eine Zielrichtung festlegt, bevor man sein schwer verdientes Geld für ein System ausgegeben hat. Zweitens, es soll Interesse an diesem Gebiet vermittelt werden, wenn noch über fundiertes Wissen verfügt wird.



More Basic Computer Games

Ahl, David H., 188 Seiten, 30 DM (in Englisch)

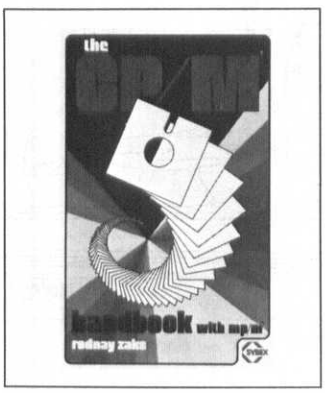
Computerspiele gewinnen immer mehr Freunde. Dieser Folgeband bietet wieder eine Menge Spaß und Anregung mit 84 brandneuen Spielen, ausgestattet mit den fantasievollsten Attributen: Mit „Camel" lernen Sie das Überleben in der Wüste, „Father" diskutiert an Ihrer Stelle mit den Kindern, die einfach nicht ins Bett gehen wollen, und durch „Millionaire" erhalten Sie Gelegenheit, Ihr Leben noch einmal zu planen.



Nichtnumerische Informationsverarbeitung

Bruderer, 190 Seiten, 39 DM

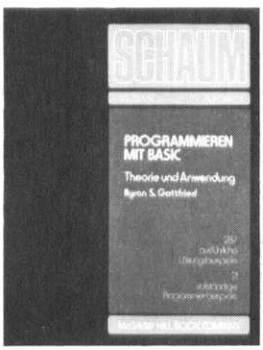
Dieses Buch wurde vom Institut für nichtnumerische Informationsverarbeitung (INIV) in der Schweiz herausgegeben und behandelt die linguistische Datenverarbeitung, künstliche Intelligenz, Computerschach, Computerkunst, automatische Dokumentation, Bibliothekautomatisierung und Rechtsinformatik.



CP/M-handbook

Von Rodnay Zaks, 322 Seiten, 39 DM (in Englisch)

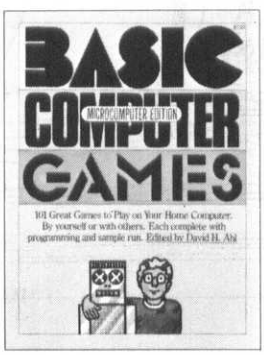
Die Bezeichnung CP/M steht für „Control Program for Microprozessors". Prinzipiell unterscheidet man in diesem Buch zwischen der Hardware (Computer, Disk, CRT und Printer) und der Software (CP/M) und MP/M). Die Beschreibung der Hardware wird mit zahlreichen Fotos unterstützt und ergänzt. Über CP/M und MP/M erfolgt die Einführung in die Arbeitsweisen der einzelnen Techniken.



Programmieren mit Basic

Gottfried, Byron, S., 238 Seiten, 27,80 DM

Der im ständigen Wechsel zwischen Theorie und praktischen Beispielen leicht verständlich aufbereitete Stoff führt in acht Kapiteln von den Grundlagen bis zu Vektor und Matrizen-Operationen. Schritt für Schritt wird die Entstehung des Programms erklärt, dazu das Flußdiagramm. Mathematisch und auch physikalisch Interessierte kommen ebenso auf ihre Kosten wie Spielprogrammfans.



Basic Computer Games

Ahl, David H., 185 Seiten, 30 DM (in Englisch)

Es darf gespielt werden – mit Basicprogrammen. Dieses Buch in englischer Sprache ist ein Dorado für Spielprogrammfans. 101 Spiele (mit Erklärung und Programmliste) verschiedener Art sorgen für Freude aller Benutzer. Jede Kategorie ist mit einer größeren Zahl vertreten: Weltraum (Super Star Trek), Sport (Bowling, Hockey), Logikspiel (Digits, Word, Tic Tac Toe), Casinospiele (Poker, Roulette), Bildungsspiele (Change). Bei diesem Buch steht der Lerneffekt nicht im Vordergrund.

6800, Programmieren in Assembler

Leventhal, 350 Seiten, 49 DM

Dieses Buch enthält eine ausführliche Beschreibung der Assemblersprache für den Mikroprozessor 6800. Es enthält zahlreiche Programmierungsbeispiele, vom einfachen Laden des Speichers bis zu kleineren Projekten. Jeder Befehl ist detailliert erklärt, sowie die einzelnen Vereinbarungen für den 6800. Hierzu gehören noch Programmierbeispiele im Standard-Format, einschließlich Flußdiagramm, Quellprogramm, Objektcode und erläuterndem Text.

Für Ihre Bestellung verwenden Sie bitte die am Heftende beigefügte Karte.

Bild 1

Bild 2

Wenn man gegenständliche Grafik mit Computer und Plotter anfertigt, besteht das Programm fast nur aus einer Unmenge von DATA. Der Computer gibt die DATA (in diesem Fall Koordinaten) an den Plotter weiter, und der Plotter besorgt den Rest. Anders ist es, wenn man diese DATA manipulieren will. Bei den „Davids" von Michelangelo ist das richtige Verhältnis zwischen X- und Y-Koordinaten verfälscht worden, so daß ein lustiger Effekt „vor, während und nach der Kur" entstanden ist. Auf den nächsten Seiten finden Sie Grafikprogramme für die verschiedensten Computer.

Shapes — Gestalt im Griff
Mehr Farbe für den Apple
Multipilotprogramm für TI-59/PC 100
Erweiterter Zeichenvorrat mit dem HP 41C
Dreidimensionale Darstellung auf dem CBM 2001
Zeichnen mit CBM Funktionen auf TRS 80

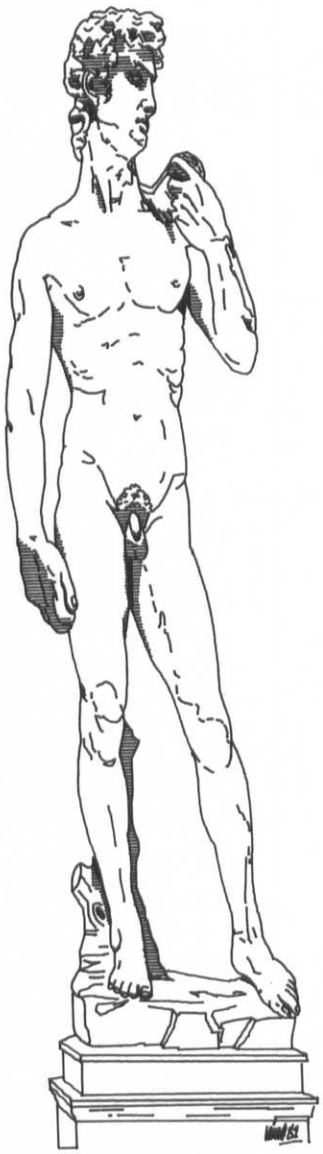


Bild 3

Bild 4

Zu den Abbildungen:

Grafik und Programm von N. G. Barbieri. Ausführung mit dem Plotter WATANABE DIGI-PLOT WX4671. Das zweite Bild von links zeigt die Grafik in der normalen Ausführung. Die anderen Bilder sind einige vom Computer geschaffene Variationen.

Nino B. Barbieri

Shapes — Gestalt im Griff

Beschreibung

Wenn es für einen normalen Sterblichen etwas Schwieriges, Verkorkstes und Kompliziertes in dem im übrigen hervorragenden Apple-Grafik-System gibt, sind es die berüchtigten Shapes.
Wenn irgend jemand mal versucht hat, Shapes nach den Instruktionen des Applesoftmanuals (siehe Kapitel „How to create a Shape table", „Saving a Shape table" und „Using a Shape table") herzustellen, der hat sicher sein blaues Wunder erlebt.
Wer es noch nicht weiß, ein Shape ist eine vorgefertigte Zeichnung, Symbol oder grafische Darstellung, die mit Applesoftbefehlen „DRAW", „XDRAW", „SCALE" und „ROT" vom Programm aus steuerbar ist. Dieses bedeutet, wenn Sie ein Raumschiff für ein Weltraumkriegsspiel auf dem Bildschirm erscheinen und bewegen lassen wollen, brauchen Sie nicht mühselig mit „HPLOT" hier und „HPLOT" dort zu operieren (damit wird Ihr Raumkriegsprogramm bestimmt viel zu langsam), sondern Sie brauchen nur das Raumschiff als „Shape" vorzufertigen, diesen sie dann mit einfachen Befehlen steuern, was einen erheblichen Gewinn an Programmgeschwindigkeit bedeutet.
Zuerst ist es notwendig, das Apple-Shape-System kurz zu umreißen: Um ein Shape zu bilden, verfügt man über acht verschiedene sogenannte Vektoren, und zwar für eine Bewegung nach oben, nach rechts, nach unten und nach links ohne PLOT, d.h. ohne daß eine Zeichnung sichtbar wird, und für dieselben vier Bewegungen, aber mit PLOT, d.h. daß eine Zeichnung sichtbar wird. Um dieses zu erreichen, haben sich die Apple-Leute ein sehr trickreiches System ausgedacht. Sie haben nämlich ein Byte genommen und dessen acht bit in drei Gruppen zerlegt, und zwar von rechts nach links erst drei und dann nochmals drei und zuletzt zwei bit. Jede dieser drei Gruppen kann einen der Vektoren enthalten, je nachdem, was für ein Binärkode hineingeschrieben wird. Jeder wird jetzt denken, daß die Sache ziemlich einfach ist. Einfach ist sie aber nicht, da verschiedene Komplikationen hinzukommen. Erstens ist es nicht möglich, einfach Nullen und Einser direkt in die Bits hineinzuschreiben, so daß die Gruppierungen von Binärkode erst in Hexadezimal bzw. in Dezimal umgewandelt werden müssen, und dieses mit einem komplizierten und umständlichen Verfahren, was seinesgleichen sucht, und zweitens ist ein Byte voller Nullen notwendig, um das Ende des Shapes zu signalisieren. Ein Vektor nach oben ohne PLOT ist mit drei Nullen dargestellt, und infolgedessen ist es scheinbar unmöglich, mehrere Vektoren nach oben ohne PLOT hintereinander zu schreiben, da, sobald der Byte voll Nullen ist, automatisch das Ende der Shapes signalisiert wird und alles, was dahinter kommt, unberücksichtigt bleibt. Ich will mich nicht weiter über diese Anleitung äußern, jeder kann es selber im Applesoftmanual sehen, und bestimmt jeder, der es probiert hat, seine Shapes nach diesem System zu bilden, hat nach viel Arbeit und Schwitzen alle möglichen Mißbildungen herausbekommen. Wilde Gerüchte kursieren, daß, seitdem der Apple auf dem Markt ist, die Klapsmühlen verstärkte Einlieferungen registrieren, von komischen Typen, die mit glasigen Augen und zitternden Händen endlose Zahlenkolonnen auf Papierblocks schreiben und dabei vor sich hin murmeln: „Nach oben mit PLOT und nach rechts ohne PLOT" und ähnlichen Blödsinn.
Aber nicht verzweifeln: Den wichtigsten Hinweis, um die Sache zu vereinfachen, findet man in demselben Applesoftmanual, wo in dem Kapitel „How to create a shape table", „Ringbuch" Seite 92 unten, in Versalien steht: „IF ALL

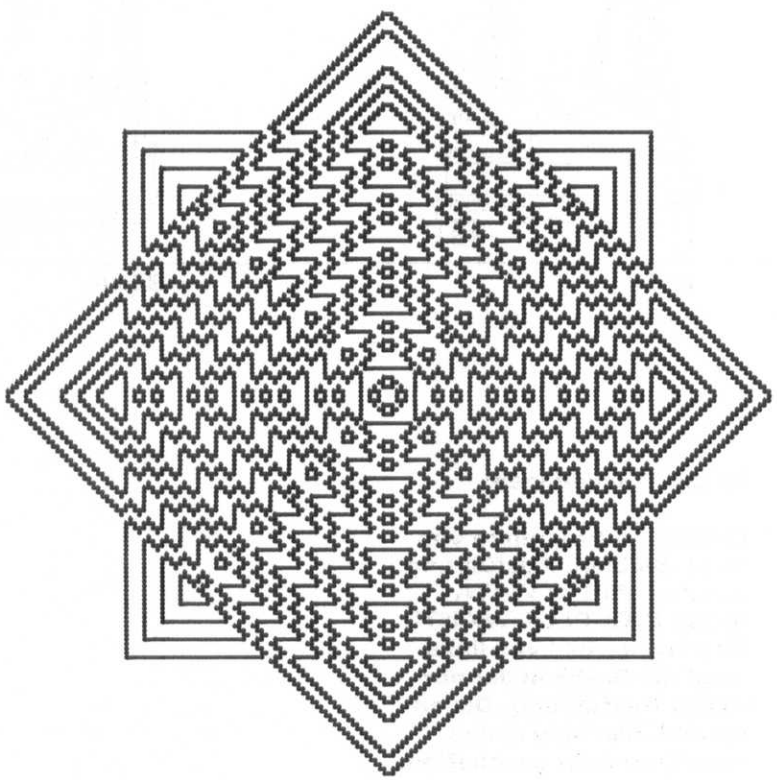


Bild 1: Eine Phase des im Text beschriebenen Programms.

Da es sich bei den im Beitrag und im Listing angegebenen Programmen um kinetische Grafiken handelt, kann man bei den Abbildungen nur die Entstehungsphase darstellen.

THE REMAINING SECTIONS OF THE BYTE CONTAIN ONLY ZEROS, THEN THOSE SECTIONS ARE IGNORED". Dieses bedeutet, daß, wenn von den drei Bitgruppen des Byte der linken (aus zwei bit) und der mittleren (aus drei bit) nur Nullen enthalten, diese überhaupt nicht ins Gewicht fallen. Also schreiben wir in jedes Byte einen einzigen Vektor und nicht mehr zwei oder drei, wie das Applesoft-Manual vorschreibt, und die Sache ist geritzt. Die Superschlauen werden sich jetzt ins Fäustchen lachen und denken, daß damit das Problem des „Schritts nach oben ohne PLOT" nicht gelöst wird, weil dadurch der ganze Byte nur Nullen enthält und damit eine Unterbrechung der Shapes verursacht wird. Aber gehen wir erst methodisch vor. Um einen oder mehrere Shapes mit einem Programm benutzen zu können, muß man drei verschiedene Dinge tun: 1. eine Shape-Datei bilden, 2. die zugehörigen Shapes schreiben und 3. diese Datei in eine geeignete Speicherstelle des RAM legen und diese Speicherstelle in die sogenannten Shape-Pointer Bytes der Zero-Page vermerken, so daß der Apple den Anfang der Shape-Datei jederzeit finden kann. Diese zwei Bytes haben die Dezimaladressen 232 und 233.

Fangen wir zuerst mit der Shape-Datei an. Von vornherein muß man wissen, wieviel Shapes die Datei enthalten wird (maximal 255 davon). Nehmen wir an, wir wollen der Einfachheit halber eine Shape-Datei, die nur drei Shapes enthalten soll, bilden. In diesem Fall wird in den ersten Byte der Datei die Zahl drei geschrieben, in den zweiten Byte immer eine Null, und dann werden sechs weitere Bytes reserviert (nämlich zwei für jeden vorgesehenen Shape, d.h., wären die vorgesehenen Shapes hundert, müßte nach 200 Bytes reservieren), und danach werden Zahlen, die die jeweiligen Vektoren der Shapes darstellen, der Reihe nach geschrieben, jedes Shape voneinander getrennt von einem Byte, der Nullen enthält. In diesen reservierten Bytes kommt der Hinweis, wo die einzelnen Shapes innerhalb der Datei anfangen. Der erste Shape fängt auf jeden Fall nach acht Bytes an (zwei Bytes sind mit 4,0 belegt, und sechs sind

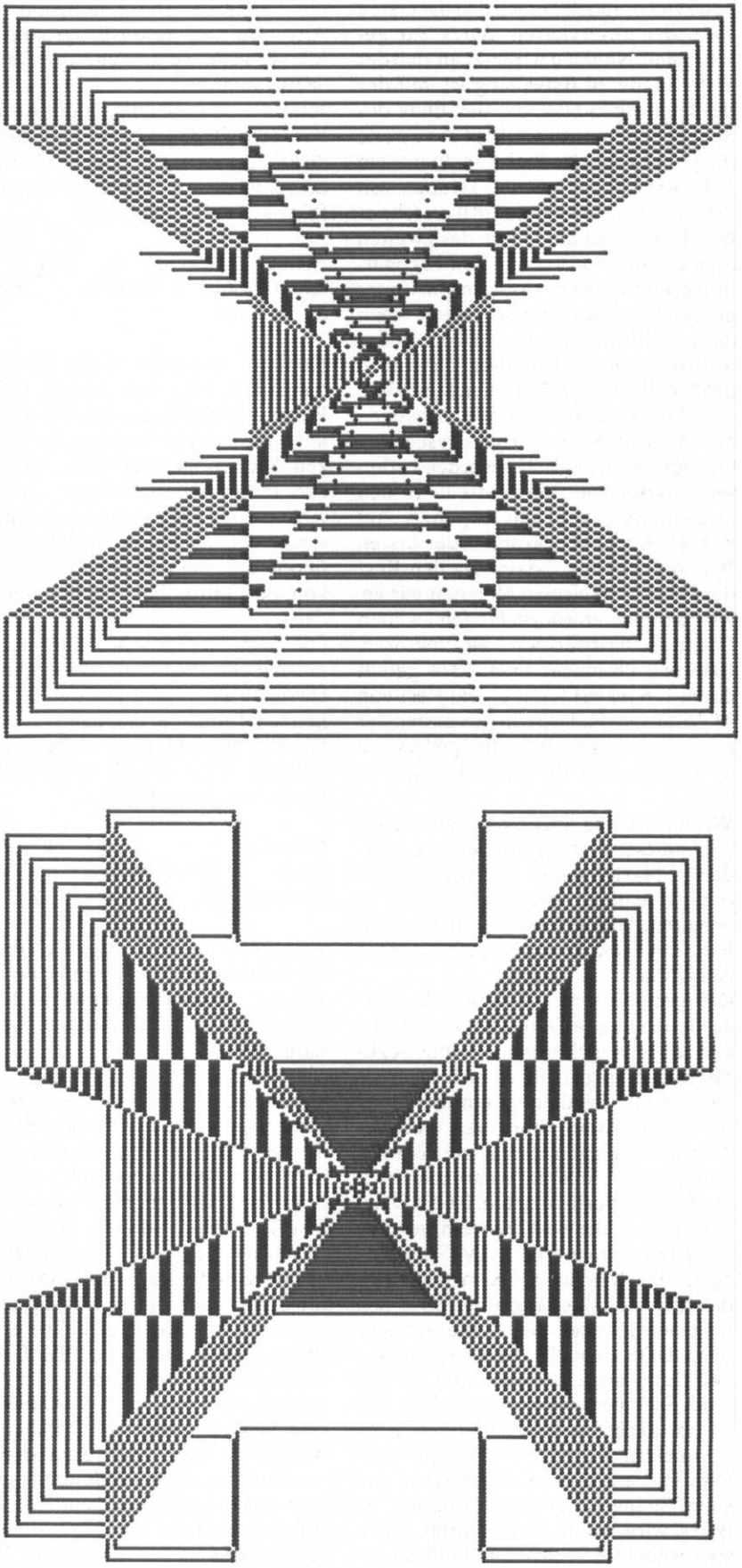


Bild 2 und 3: 2 Phasen des Programms „Alien Heart".

Shapes — Gestalt im Griff

reserviert), und daher wird in die ersten zwei der reservierten Bytes 8,0 geschrieben. Nehmen wir jetzt an, daß der erste Shape 20 Bytes lang ist, mit der dazugehörigen Null, die das Ende des Shapes bedeutet, dann fängt der zweite Shape nach der 28. Stelle an. In diesem Falle wird in der zweiten Gruppe von den reservierten Bytes 28,0 geschrieben. Dann wird die Länge des zweiten Shapes hinzuaddiert und das Ergebnis in die letzten zwei reservierten Bytes geschrieben. Was passiert aber, wenn diese Zahl größer als 255 ist? (Jeder weiß nämlich, daß es unmöglich ist, eine größere Zahl als 255 in einen 8-bit-Byte zu schreiben). Aber gerade deswegen haben wir je Shape 2 Bytes reserviert. Größere Zahlen als 255 werden in diesen 2 Bytes wie folgt verteilt: In den zweiten Byte wird das Ergebnis von Zahl geteilt durch 256 und in den ersten Byte das Modulo davon, d.h. den Rest der Division. Beispiel: Nehmen wir an, wir haben einen enorm langen zweiten Shape geschrieben von 1346 Vektoren. 1346 und 28 macht 1374. 1374 geteilt durch 256 macht 5, mit einem Rest von 94. In diesem Fall wird in den ersten der zwei reservierten Bytes die Zahl 94 und in den zweiten die Zahl 5 geschrieben.

Wie macht man jetzt einen der Shapes, die in dieser Datei enthalten sein werden? Nach dem Barbieri-System braucht man nur folgendes im Kopf zu behalten: Uhrzeigersinn und die Zahlen 1 bis 7 und 88 dazu. Auch hierüber Ausführlicheres später.

Die geeignetsten RAM-Speicherstellen für die Shape-Datei sind vom Rechner und dazugehörigen Peripheriegeräten abhängig. Falls es sich um eine kurze Datei handelt, so um die 80, 90 Bytes insgesamt, kann man diese ab Adresse 768 legen, da gerade dort der Apple einen sicheren, wenn auch kleinen Ort zur Verfügung hat. Ist die Datei länger, ist es das beste System, von der höchsten verfügbaren RAM-Speicherstelle (ins Manual schauen bezüglich dieser Stelle in Abhängigkeit, ob es sich um einen 32- oder 48K Apple, mit oder ohne DOS handelt) die Gesamtlänge der Datei abziehen, das HIMEM: an diese Speicherstelle setzen und die Datei am nächsten Byte anfangen. Zum Beispiel: Im Falle eines 48K Apple mit Disk-Drive, also mit DOS im RAM und einer Shape-Datei von insgesamt 98 Bytes, wird wie folgt verfahren: höchste verfügbare Speicherstelle 38399 minus Länge der Datei 98 macht 38301. Also im Applesoft-Programm HIMEM: 38301 setzen und die Datei ab 38302 anfangen.

Jetzt ist es notwendig, in die Adressen der Zero-Page 232 und 233 die Zahl 38301 zu vermerken. Wer es nicht weiß, dieses wird vom Applesoft-Programm folgendermaßen gemacht:

```
POKE 232,38302 — 256 * INT (38302/256): POKE 233,INT (38302/256)
```

Kommen wir jetzt zu der Shape-Bildung. Erst muß man wissen, daß der Punkt, wo man jeden einzelnen Shape anfängt, auch der Punkt ist, der die ganzen Shapes nachher vom Applesoft-Programm steuert. Nehmen wir an, wir wollen ein Quadrat zeichnen, das hinterher von der mittleren Achse aus rotieren soll, muß dieses Quadrat auch von der Mitte her angefangen werden.

Die verschiedenen Vektoren, jeweils mit und ohne PLOT, mit dazugehörigen Dezimalzahlen sind aus der untenstehenden Tabelle ersichtlich:

Die Zeichen bedeuten: → Bewegungsrichtung ohne PLOT und ↔ Bewegungsrichtung mit PLOT.

Vektoren	↑	↗	↓	←	↕	↔	↕	↔
Binär	01011000	1	10	11	100	101	110	111
Dezimal	88	1	2	3	4	5	6	7

Also leicht zu merken, im Uhrzeigersinn werden die Vektoren ohne PLOT mit den Dezimalzahlen 88, 1, 2 und 3 und die mit PLOT mit 4, 5, 6 und 7 dargestellt.

Sie werden jetzt bestimmt fragen, warum überhaupt 88 eine Bewegung nach oben, ohne PLOT darstellen soll. Sie tut es eben nicht, sondern der Reihe nach eine Bewegung nach oben, nach links und wieder nach rechts ohne PLOT (die drei bit-Gruppen: 01 011 000 von rechts nach links gelesen bedeuten eben, einmal die 0, einmal die 3 und dann die 1). Mit diesem System ist es möglich, mehrere leere Schritte nach oben, ohne PLOT zu machen, ohne daß ein Byte voller Nullen wird und damit unseren schönen Shape plötzlich unterbricht. Infolgedessen wird unser von der Mitte aus steuerbares Quadrat mit einer Seitenlänge von vier Vektoren mit folgender Zahlenreihe dargestellt:

2,2,7,7,4,4,4,4,5,5,5,5,6,6,6,6,7,7,0. Dieses bedeutet der Reihe nach zwei Schritte ohne PLOT nach unten, zwei mit PLOT nach links, vier mit PLOT nach oben, vier mit PLOT nach rechts, vier mit PLOT nach unten, zwei mit PLOT nach links, um das Quadrat zu schließen, und dann eine Null, um das Ende des Shapes zu signalisieren.

Nehmen wir an, daß wir ein kleines Applesoft-Programm schreiben wollen, welches dieses Quadrat als Shape in einer gegebenen Größe rotieren läßt. Dieses Programm sieht so aus:

```
10 HIMEM: 38376
```

Dieses ist nämlich die höchste RAM-Stelle minus Länge der Shape-Datei, 23 Bytes, wie wir später sehen werden.

```
20 FOR I = 38377 TO 38399
```

Hier fängt eine Schleife an, die alle Adressen zwischen den zwei Zahlen durchgeht.

```
30 READ A: POKE I,A
```

Diese Zeile liest der Reihe nach die Werte in der untenstehenden Data und schreibt der Reihe nach diese Werte in die jeweiligen Adressen.

```
40 NEXT I
```

Diese Zeile schließt die Schleife.

```
50 DATA 1,0,4,0
```

Diese vier Zahlen bedeuten der Reihe nach: Diese Datei enthält ein einziges Shape, die übliche Null, die immer an dieser Stelle erscheinen muß, und die letzten zwei Zahlen, daß dieser Shape nach dem vierten Byte anfängt.

```
60 DATA
2,2,7,7,4,4,4,4,5,5,5,5,6,6,6,6,7,7,0
```

Sie werden schon erkannt haben, daß obige Zahlen unser Quadrat darstellen.

```
70 POKE 232,38376—256*INT (38376/256):POKE233, INT (38376/256)
```

Hiermit wird der Shape-Datei-Pointer gesetzt.

80 HGR2

HIRES-Grafik-Mode einschalten.

90 R = 0: S = 20

Mit der Variable R werden wir die Rotation des Quadrates steuern. Die Variable S ist für SCALE zuständig.

100 ROT = R: SCALE = S

Jetzt ist die Rotation gleich R, d.h. Null. Der SCALE, d.h. die Vergrößerung ist 20mal die 4 Punkte der Seitenlänge des ursprünglichen Quadrats, also 80 Punkte Seitenlänge.

110 XDRAW 1 AT 139,95

Dieser Befehl zeichnet unser Quadrat in die Mitte des Bildschirms.

120 XDRAW 1 AT 139,95

Damit wird das Quadrat wieder verschwinden.

130 R = R + 1

Damit wird R und somit auch die Rotation um den Wert eins gesteigert.

140 IF R = 64 THEN R = 0

Hiermit verhindern wir, daß die Rotation größer als 64 wird und nach Möglichkeit, im Laufe der Zeit, größer als 255 mit ILLEGAL QUANTITY ERROR.

150 GOTO 100

Alles von vornherein noch einmal, aber mit dem Quadrat um einen Wert gedreht.
Dieses Programm wird mit CTRL-C gestoppt.
Wer aus diesem einfachen Programm eine Computer-Grafik mit Bewegung herstellen möchte, der kann die Zeile 90 und die Zeile 130 wie folgt umändern:

```
 90 R = 0: S = 1
130 R = R + 8
```

und diese zwei Zeilen neu hinzufügen:

```
114 S = S + 1: IF S > 30 THEN S = 1
116 SCALE = S
```

Der Leser kann auch weiter mit anderen geometrischen Figuren, mit anderen Parametern oder auch mit Random-Zahlen experimentieren. Der Fantasie ist keine Grenze gesetzt.
Jetzt, wo wir wissen, wie einfach das Ganze funktioniert, können wir zu ernsthafteren Programmen übergehen.
Ein sehr reizvolles, mit obigem System geschriebenes Programm ist in folgendem Listing dargestellt. Geometrische Formen mit Animation und Tiefenwirkung werden auf dem Bildschirm lebendig, mit einer Unzahl von Variationen.

Viel Spaß!

```
ÜLIST

10  REM  ALIEN HEART
20  REM  COMPUTER-ART
30  REM  BY N. G. BARBIERI-DUESSELDORF
40  HIMEM: 38130
45  REM  DIE NAECHSTEN ZWEI ZEILEN BILDEN DIE SHAPE DATEI
      AB ADRESSE 38131
50  FOR I = 38131 TO 38399
60  READ A: POKE I,A: NEXT
65  REM  IN ZEILE 70 WIRD DER POINTER ZU ANFANG DER SHAPE
      DATEI GESETZT
70  POKE 232,38131 - 256 *  INT (38131 / 256): POKE 233, INT
     (38131 / 256)
90  REM  ZEILE 100 BEDEUTET, DASS DIE DATEI 9 SHAPES ENTH
    AELT, DIE JEWEILS AN DEN NACHFOLGENDEN STELLEN ANFAN
    GEN
100  DATA 9,0,20,0,46,0,65,0,78,0,101,0,144,0,186,0,210,0
    ,244,0
105  REM  DIE FOLGENDEN 9 DATA-REIHEN STELLEN 9 GEOMETRIS
    CHE FIGUREN DAR
110  DATA  2,7,6,7,4,5,4,4,7,4,5,6,5,5,4,5,6,7,6,6,5,6,7,
    4,7,0
120  DATA  2,2,7,4,7,4,4,5,4,5,5,6,5,6,6,7,6,7,0
130  DATA  2,7,7,4,4,5,5,5,5,6,6,7,7,0
140  DATA  2,2,7,6,7,4,4,4,4,5,6,5,5,4,5,6,6,6,6,7,4,7,0
150  DATA  2,2,7,7,7,6,6,5,5,4,4,4,4,4,4,7,7,6,6,5,5,5,5,
    5,5,4,4,7,7,6,6,6,6,6,6,5,5,4,4,7,7,7,0
160  DATA  2,2,3,7,7,6,6,5,5,4,4,88,88,4,4,7,7,6,6,5,5,1,
    1,5,5,4,4,7,7,6,6,2,2,6,6,5,5,4,4,7,7,0
170  DATA  2,2,2,7,4,7,7,4,4,5,5,4,5,5,6,5,5,6,6,7,7,6,7,
    0
180  DATA  2,2,3,7,7,6,5,5,4,88,88,4,7,7,6,5,5,1,1,5,5,4,
    7,7,6,2,2,6,5,5,4,7,7,0
190  DATA  2,2,2,7,4,3,7,4,4,5,1,4,5,5,6,1,5,6,6,7,3,6,7,
    0
200  HGR2
210 A = 1:B = 2:C = 30: REM  3 VARIABEL FUER DIE DARAUF F
    OLGENDE SCHLEIFE
215  REM  ZEILE 220 RANDOMISIERT SHAPE UND ROTATION - DIE
     SHAPE ROTATION KANN ENTWEDER 0 ODER 16, D. H. NORMA
    L ODER UM 90 GRAD GEDREHT SEIN
220 SH =  INT (9 *  RND (1) + 1): ROT=  INT (2 *  RND (1)
    ) * 16
230  FOR I = A TO C STEP B
240  SCALE= I: XDRAW SH AT 139,95: NEXT
245  REM  ZEILE 250 KEHRT DIE SCHLEIFE PARAMETER UM, SO D
    ASS DER NAECHSTE SHAPE GROSS ANFAENGT UND KLEINER WI
    RD
250  IF B = 2 THEN B =  - 2:A = 30:C = 1: GOTO 230
260  GOTO 210
```

Nino B. Barbieri

Mehr Farbe für den Apple

Beschreibung:

Mit einem kleinen Kunstgriff kann man mehr als 20 Farbschattierungen erzeugen anstatt der üblichen vier + Schwarz und Weiß der HIRES-Grafik des Apple II.

Ich entdeckte, daß, wenn ich eine horizontale Linie in der Farbe A, direkt darunter gefolgt von einer horizontalen Linie in der Farbe B plottete, das Ergebnis eine dickere Linie in einer neuen Farbe C war. Durch Probieren kam auch heraus, daß, wenn ich zuerst die Farbe B und dann darunter die Farbe A legte, das Ergebnis eine neue Farbe D war. Z.B. werden mit den HIRES-Farben 2 und 5 einmal Rosarot und einmal Flieder erzeugt.

Wer also mit der Apple-HIRES-Grafik Flächen in anderen, normalerweise nicht vorhandenen Farben erzeugen will, der braucht nur, alternativ, horizontale Linien in verschiedenen Farben plotten zu lassen. Dieses klappt nur mit horizontalen Linien, mit vertikalen Linien ist es nicht möglich, aus Gründen, die hier zu erklären zu weit führen würden.

Aus welchen Farbkombinationen die neuen Farben zu erzeugen sind, kann jeder selber entdecken mit dem kleinen Programm des Listings Nr. 1. Dieses Programm fragt nach einer ersten und zweiten Farbe (anzugeben mit den Zahlen von 0 bis 7), und dann bilden sich auf dem Bildschirm vier Vierecke, und zwar der Reihe nach: das erste in der zuerst definierten Farbe, das zweite in der zweiten angegebenen Farbe, das dritte in der Farbmischung: erst die erste Farbe und dann die zweite Farbe, und das vierte in der Farbmischung: zuerst die zweite Farbe und dann die erste Farbe.

Listing Nr. 2 ist ein Computergrafik-Programm, das den obenbeschriebenen Kunstgriff ausnutzt. Alle möglichen HIRES-Farbkombinationen werden auf dem Bildschirm sichtbar. Die REMS erklären ausführlich den Programmverlauf.

ÜLIST

Listing Nr. 1:

```
10  REM  FARBTEST
20  REM  SONDERFARBEN IN HIRES-GRAFIK
30  REM  VON N.G. BARBIERI - DUESSELDORF
50  HOME : HGR
60  HOME : VTAB 21: INPUT "1. FARBE NR. : ";A
65  IF A > 7 OR A < 0 THEN 60
70  VTAB 21: HTAB 19: INPUT "/  2. FARBE NR. : ";B
75  IF B > 7 OR B < 0 THEN 70
78  REM  ZEILEN 60 - 75 SIND INPUT VON ERSTER UND ZWE
    ITER FARBE
80  FOR I = 41 TO 120 STEP 2
90  HCOLOR= A
100  HPLOT 0,I TO 59,I
110  HPLOT 0,I + 1 TO 59,I + 1
120  HCOLOR= B
130  HPLOT 70,I TO 129,I
140  HPLOT 70,I + 1 TO 129,I + 1
150  HCOLOR= A
160  HPLOT 140,I TO 199,I
170  HCOLOR= B
180  HPLOT 140,I + 1 TO 199,I + 1
190  HCOLOR= B
200  HPLOT 210,I TO 269,I
210  HCOLOR= A
220  HPLOT 210,I + 1 TO 269,I + 1
230  NEXT I
235  REM  ZEILE 80 - 230 BILDEN DIE 4 VIERECKE, 2 IN
    DER INPUT-FARBE UND 2 IN DEN FARBMISCHUNGEN
240  VTAB 24: PRINT "DRUECKE IRGENDEINE TASTE ";: GET
    A$
245  REM  NACH BETAETIGUNG IRGENDEINER TASTE KANN MAN
     DIE KOMBINATION ZWEIER NEUER FARBEN TESTEN
250  GOTO 60
```

ÜLIST

Listing Nr. 2:

```
10  REM  GABY
20  REM  COMPUTER-ART
30  REM  VON N.G. BARBIERI
40  HOME : HGR2
45  REM  HIRES-GRAFIK EINSCHALTEN
50  FOR I = 1 TO 55
55  REM  ES SOLLEN 55 VIERECKE VERSCHIEDENER GROESSE U
    ND FARBE GEPLOTTET WERDEN
60 C1 =  INT (6 *  RND (1) + 1): IF C1 = 3 OR C1 = 4 THEN
    60
65  REM  HIER WIRD EINE ERSTE RANDOM-FARBE DEFINIERT
70 C2 =  INT (6 *  RND (1) + 1): IF C2 = 4 THEN 70
75  REM  HIER WIRD EINE ZWEITE RANDOM-FARBE DEFINIERT
80 A =  INT (279 *  RND (1)):B =  INT (279 *  RND (1))
    : IF A = B THEN 80
85  REM  A + B SIND ZWEI RANDOM X-KOORDINATEN
90 C =  INT (191 *  RND (1)):D =  INT (191 *  RND (1))
    : IF C = D THEN 90
95  REM  C + B SIND ZWEI RANDOM Y-KOORDINATEN
100 F = 3: IF C > D THEN F =  - F
105  REM  FALLS DIE Y-KOORDINATE C GROESSER ALS DIE Y-
    KOORDINATE D IST, SOLL DIE FOLGENDE SCHLEIFE NEGA
    TIV VERLAUFEN
110  FOR J = C TO D STEP F
115  REM  DIESE SCHLEIFE PLOTTET DIE EINZELNEN RANDOM-
    VIERECKE
120  HCOLOR= C1: HPLOT A,J TO B,J
125  REM  ERST EINE LINIE IN DER FARBE C1...
130  HCOLOR= C2: HPLOT A,J + 1 TO B,J + 1
135  REM  .... UND DANN EINE LINIE IN DER FARBE C2. DI
    E DRITTE LINIE WIRD NICHT GEPLOTTET, UM DIE SACHE
     NOCH REIZVOLLER ZU GESTALTEN
140  NEXT J: NEXT I
145  REM  DIE OBIGEN SCHLEIFEN J UND I WERDEN HIER GES
    CHLOSSEN
150  FOR H = 1 TO 3333: NEXT H
155  REM  EINE LEERE SCHLEIFE, UM DAS FERTIGE "KUNSTWE
    RK" ZU GENIESSEN
160  GOTO 40
165  REM  ZURUECK ZUM ANFANG, UM EIN GANZ NEUES BILD Z
    U ERHALTEN
```

Marius Heyn

Erweiterter Zeichenvorrat mit dem HP 41 C

Beschreibung:

Nachfolgende Manipulation beruht im wesentlichen darauf, daß aus dem HP 41 C entfernte Module ihre Daten etwa 25 s behalten und eine Umwandlung von Programmspeicherinhalten in Datenspeicher bei angeschlossenen Modulen nicht möglich ist (bei Ausführung von SIZE — — — erscheint dann „TRY AGAIN" in der Anzeige). Wäre dies, wie beim TI 59 möglich, könnte man aus den gespeicherten Programmen durch SIZE Daten generieren; dies verhindert jedoch das Betriebssystem.

Beispiel:

Ausgangskonfiguration: HP 41 C, Kartenleser, 2 Erweiterungsmodule.
Vorgehen:

1. SIZE 000 einstellen
   Programmspeicher mit verschiedenen Programmen füllen
   letzter Stand, z.B. 00 REG 03
   Rechner ausschalten
2. Kartenleser und beide Module kurzzeitig entfernen (etwa 10 s)
   Rechner einschalten (Module und Kartenleser noch entfernt!)
   Anzeige: MEMORY LOST
3. Rechner ausschalten, Module und Kartenleser wieder einstecken
   Rechner einschalten, PRGM drükken: 00 REG 46 (Grundeinstellung 17 Datenregister, 46 Programmregister).

Die Register 18 bis 144 erhalten jetzt Zeichenketten unterschiedlicher Länge, die meist auch neue Zeichen und Ziffern beinhalten. Im Beispiel wurden auch verstümmelte Teile von alphanumerischen Abfragen, die in den eingegebenen Programmen enthalten waren, wiedergefunden. Weiterhin enthalten die Register Dezimalzahlen mit und ohne Exponent.
Register mit Adressen über 99 werden dabei folgendermaßen aufgerufen: z.B. 102 (X-Register), RCL IND . X.
Die Zeichenketten mit neuen Zeichen wurden in die Register ab 00 abgespeichert und mit WDTAX aufgezeichnet.

```
R00= "UwuU]↑
"
R01= "("
R02= "]u→_wΣ
"
R03= "UU⊢]w\
"
R04= "UUU_WU
"
R05= "x̄" x̄"×
"
R06= "Ä@"
R07= "Ä@"
R08= "Ä@"
R09= "αQβAΩA
"
R10= "βEβ@βQ
"
R11= "××ββΩ@
"
R12= "αββ××¨
R13= "=?μ↾Üμ
"
R14= "=?μ↾Üμ
"
R15= "Cᵀβ@x̄t
"
R16= "&×1"
R17= "δ"
R18= "↾ΩȦ←#ᵀ
```

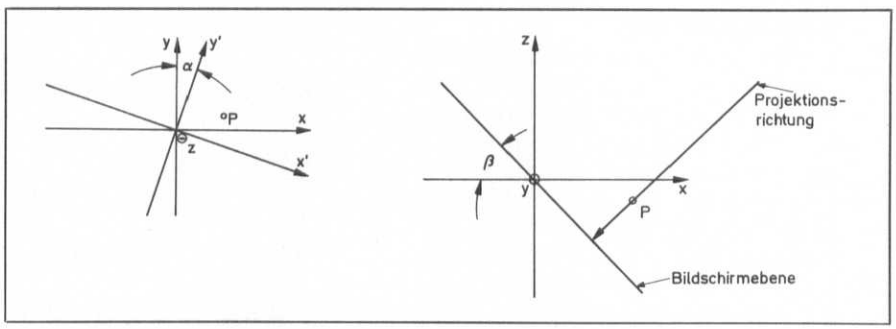
Durch ARCL NN (NN = Registeradresse mit gewünschten Zeichen) können im Alphamodus Korrekturen an dem String vorgenommen werden. Die erhaltenen Daten lassen den Schluß zu, daß hier eine ähnliche Umwandlung erfolgt ist, wie sie beim TI 59 erfolgt, wenn aus dem Programmspeicherbereich problemlos (was auch gefährlich ist, wenn dann die Register aus Versehen gelöscht werden) Datenspeicher gemacht werden kann.

Frank Wähner

Dreidimensionale Darstellung auf dem CBM 2001

Beschreibung

Dieses Programm ermöglicht die Darstellung räumlicher Gebilde (z.B. dreidimensionale Funktionen, Würfel, usw.) auf dem Bildschirm. Geeignet ist es für den CBM 2001 bzw. nach Änderungen für 6502-Computer allgemein. Während der Anzeige ist eine Drehung des Körpers um zwei Achsen möglich.



Arbeitsweise des Programms

Optimal für das Programm wäre ein Grafikinterface mit einer Auflösung von 256 × 256 Bildpunkten. Um die schlechte Auflösung des PET ein wenig zu verbessern, wurde vom Verfasser vor längerer Zeit ein kleines Unterprogramm entwickelt, das dem CBM zu einer Auflösung von 4000 Bildpunkten verhilft.
Speicherbelegung des Unterprogramms:

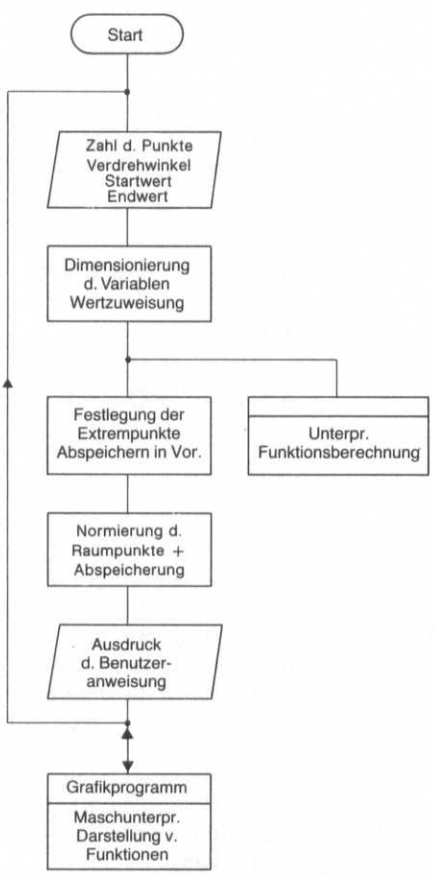
992: Spalte (0—79)
993: Zeile (0—49)
994: Punkt setzen/löschen (0/15)
995: soll vorhandenes Charakterzeichen (z.B. Buchstabe) auf Bildschirm überschrieben werden nein/ja (0/1)
996: automatisches Spalteninkrement (0...255)
997: automatisches Zeileninkrement (0...255)

Aufgerufen wird das Programm durch SYS826. Eine Fehlbedienung ist nicht möglich.
Folgende Abbildungsvorschrift (vom Verfasser entwickelt) wird verwendet:

$Y' = \cos\beta(\mathrm{COS}\alpha \times P_2 - \mathrm{SIN}\alpha \times P_1) + \mathrm{SIN}\beta \times P_3$
$X' = \mathrm{COS}\alpha \times P_1 + \mathrm{SIN}\alpha \times P_2$

Erklärung der einzelnen Parameter:
Der räumliche Körper befindet sich zusammen mit den dazugehörigen Koordinatenachsen (X,Y,Z) fest im Raum. Ein Punkt P des Körpers hat die Koordinaten P_1, P_2, P_3 (abgespeichert im Datenfeld des Maschinenprogramms). Die Bildschirmebene hat die beiden Koordinatenachsen X' und Y' und liegt zu Beginn in der XY-Ebene.

Flußdiagramm zum Basicprogramm



Erklärung der Winkel:
Die beiden Winkel α und β können nun (während die Anzeige läuft) verändert werden. Dadurch kann jeder Betrachtungswinkel eingestellt werden. Ist die Zahl der Raumpunkte klein (bis etwa 30), so ergibt dies eine kontinuierliche Drehung des Raumgebildes (Kinoeffekt).
Um die Zahl der erforderlichen Stützpunkte klein zu halten (Rechenzeitersparnis), werden diese Punkte mit Hilfe des Unterprogramms „Vektorgenerator" durch Geraden verbunden.
Hauptproblem bei der Entwicklung dieses Programmes stellte (wie schon angedeutet) die Rechenzeit dar, denn es werden bei jeder Änderung des Betrachtungswinkels (z.B. um 10 Grad) sehr umfangreiche Berechnungen durchgeführt.

Bei 20 Stützpunkten sind das:

2 Berechnungen des SIN
2 Berechnungen des COS
20 × 6 = 180 Multiplikationen
20 Divisionen
Additionen und Vergleiche nicht mitgerechnet

Das Programm benötigt dazu etwa 100 ms
Der Basicinterpreter hingegen benötigt (zum Vergleich) etwa 30 ms für die Berechnung eines einzigen SIN.

Aus diesem Grund mußten eigene schnelle Rechenroutinen geschrieben werden. Die höhere Geschwindigkeit wurde hauptsächlich durch folgende Maßnahmen erreicht:

1. Reduzierung der Genauigkeit
2. Keine Reihenentwicklung bei Berechnung der Winkelfunktionen

zu 1. Multiplikation: 8 bit × 8 bit mit 16 bit Ergebnis

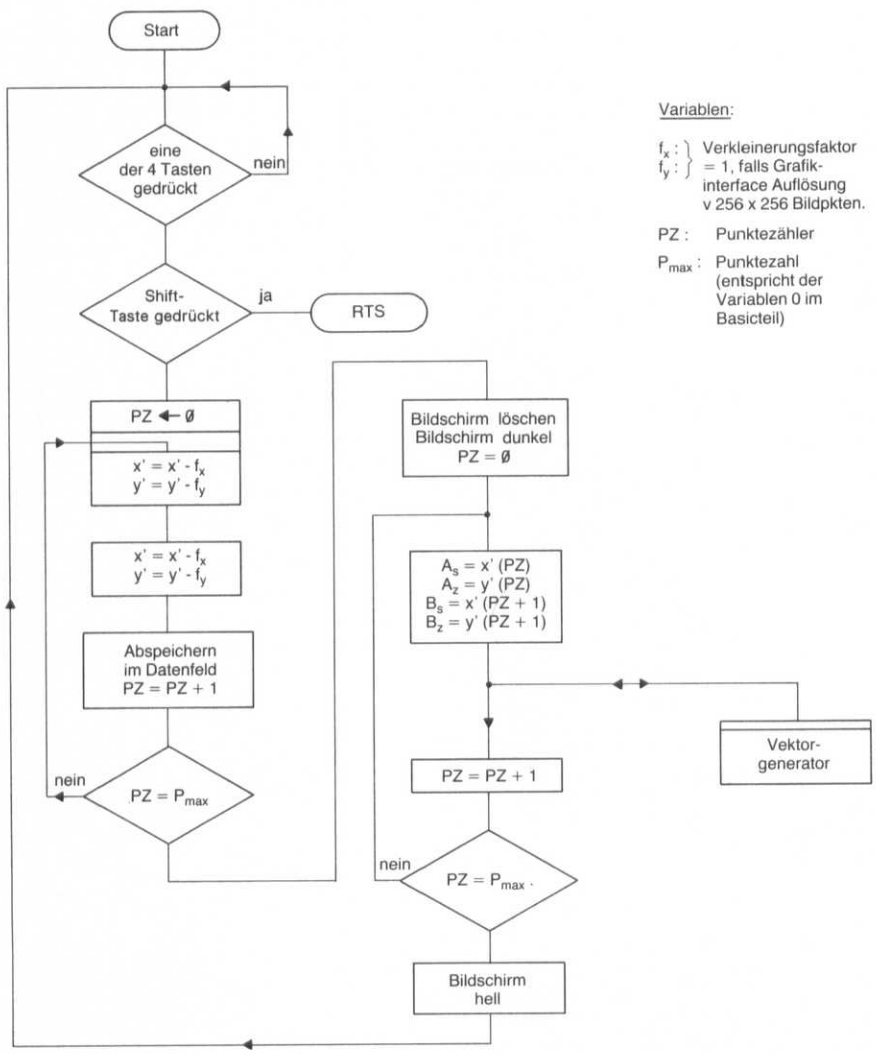
Division: 16 bit/16 bit mit 16 bit Ergebnis

Addition: 8 bit + 8 bit (für Schutzstelle)

Substr.: 8 bit + 8 bit (für Schutzstelle)

zu 2. Die Winkelfunktionen (Genauigkeit 7 bit + 1 bit f. Vorzeichen) werden durch Tabellenverarbeitung berechnet. Dadurch extrem hohe Rechengeschwindigkeit (etwa 32 µs).

Flußdiagramm zu Grafikprogramm



Variablen:

f_x : } Verkleinerungsfaktor
f_y : } = 1, falls Grafik-interface Auflösung v 256 x 256 Bildpkten.

PZ : Punktezähler

P_{max} : Punktezahl (entspricht der Variablen 0 im Basicteil)

Anwendung des Programms:

1. Laden des Maschinenteils durch Shift-RUN
2. Laden des Basicteils durch Shift-RUN

In Programmzeile Nr. 5 steht die Funktion (in Parameterdarstellung). Sie kann leicht durch eine neue Funktion ersetzt werden, z.B.

```
5 X=COS(T):Y=SIN(T):Z=T/10
```

ergibt z.B. eine Spirale

Beispiel: nach RUN
Zahl der Punkte ? 20
Verdrehwinkel/Bild ? 20 (Grad)
Untere Grenze ? 0
Obere Grenze ? 10

Nach kurzer Rechenzeit wartet der Rechner auf eine weitere Eingabe (Erklärung wird vorher ausgegeben).
Nach dem Drücken einer der 4 Tasten erscheint ein spiralenförmiges Gebilde, das sich relativ schnell (180 Grad/s) dreht, sofern man eine der Tasten gedrückt hält.

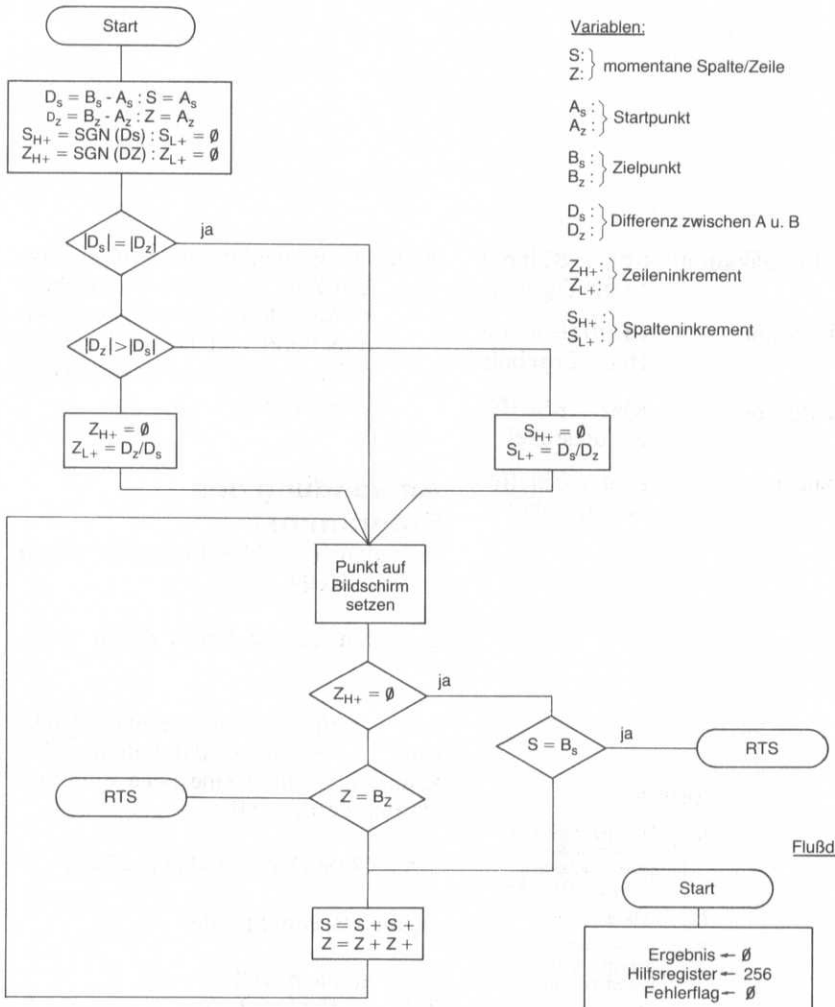
Erweiterungen:

Für einen einigermaßen erfahrenen Basic-Programmierer dürfte es nicht schwer sein, das Programm nach seinen eigenen Bedürfnissen umzuschreiben.
Vom Verfasser bereits verwirklicht:
die Darstellung zweiparametriger Funktionen (z.B. „Hüte"), wie man sie auch oft in den Anzeigen einiger Computerhersteller sieht.

(bitte umblättern)

Dreidimensionale Darstellung auf dem CBM 2001

Unterpr. Vektorgenerator



Variablen:

S: } momentane Spalte/Zeile
Z: }

A_S : } Startpunkt
A_Z : }

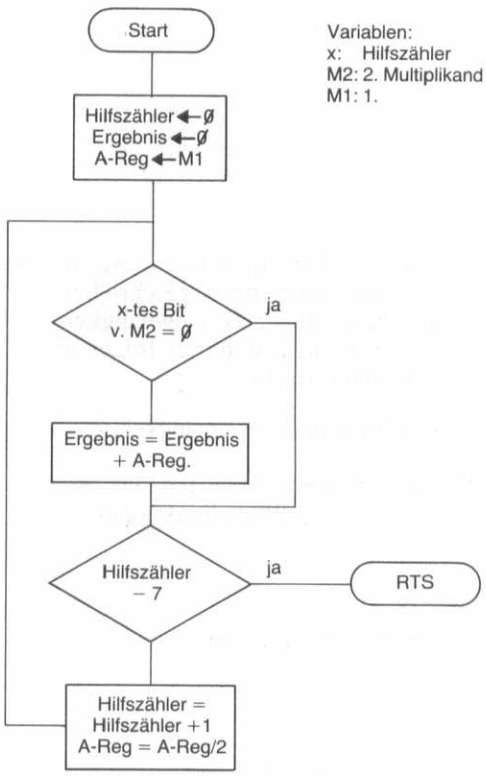
B_S : } Zielpunkt
B_Z : }

D_S : } Differenz zwischen A u. B
D_Z : }

Z_{H+} : } Zeileninkrement
Z_{L+} : }

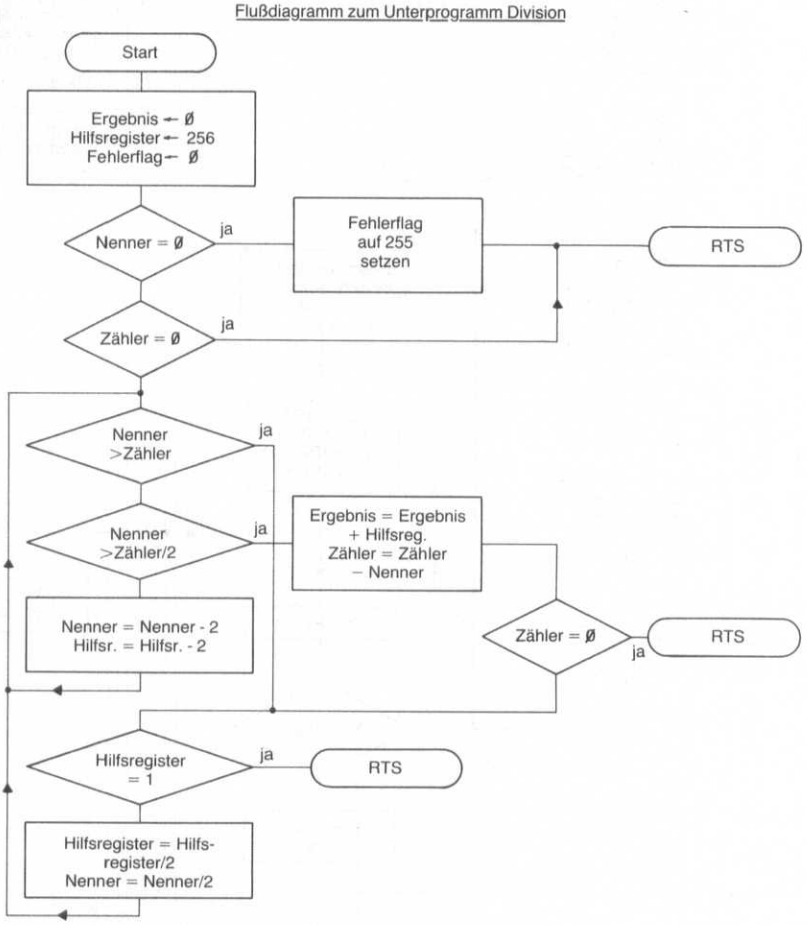
S_{H+} : } Spalteninkrement
S_{L+} : }

Unterprogramm Multiplikation



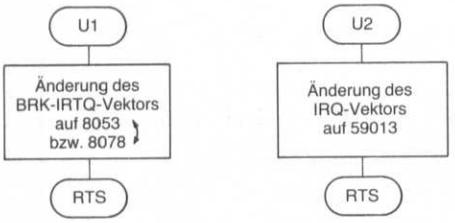
Variablen:
x: Hilfszähler
M2: 2. Multiplikand
M1: 1.

Flußdiagramm zum Unterprogramm Division



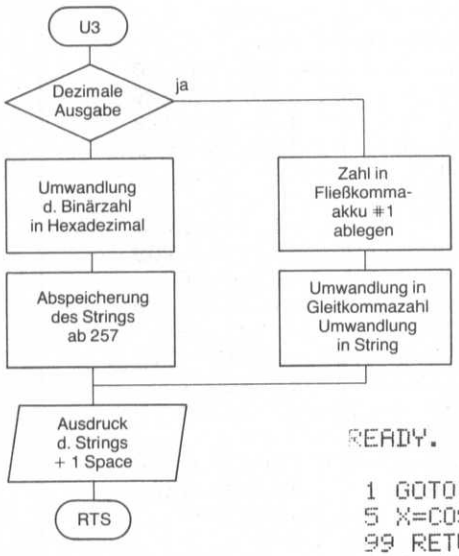
Korrekterweise muß allerdings dazu gesagt werden, daß die Auflösung der CBM'-Computer dafür nicht prädestiniert ist.
Auch die Darstellung von Würfeln, Brücken und anderen Gegenständen ist ohne weiteres möglich.
Dabei muß folgendes beachtet werden:

1. Die Stützpunkte (z.B. Eckpunkte eines Würfels) werden in den Variablen X(), Y(), Z() abgespeichert.
2. Die Variable Q steht für die Zahl der Stützpunkte.
3. Die Normierungsroutinen (ab 190) müssen unbedingt durchlaufen werden.



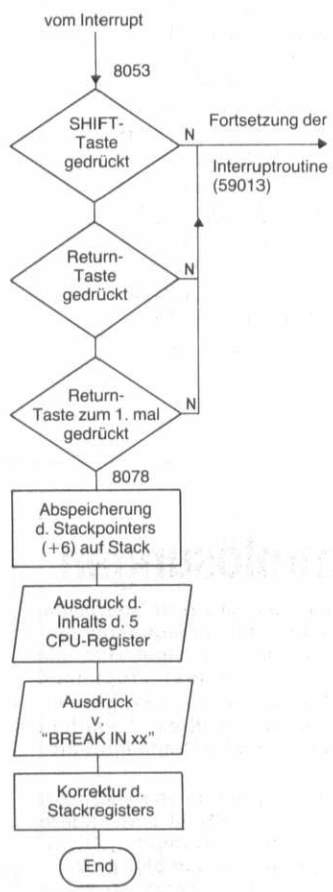
Speicherverteilung:

1024—5818 : Basicprogramm + Daten
5819—6074 : Y
6075—6330 : X
6331—6586 : Z()
6587—6842 : Y()
6843—7098 : X()
7099—7121 : Daten für Vektorgenerator
7122—7264 : Daten für Grafikprogramm
7265—7683 : Vektorgenerator (Pr.)
7684—8191 : Grafikprogramm



Änderungen für CBM (neues Betriebssystem):

Adresse:	PET	CBM	Erklärung
7685	515	151	Tastaturspeicher
8155	516	152	Flag für Shift-Taste (1, wenn gedrückt)
8172	57910	57897	Bildschirm löschen
Basiczeile	500 : Poke	525,0 (PET)	Poke 158,0 (CBM)




```
READY.

 1 GOTO100
 5 X=COS(T)*T:Y=SIN(T)*.7*T:Z=T
 99 RETURN
 100 INPUT"■ZAHL DER PUNKTE (2-230) :";Q:Q=Q-1:IFQ>229ORQ<1THEN100
 101 POKE7123,Q:DIMX(Q),Y(Q),Z(Q)
 110 INPUT"VEDREHWINKEL/BILD (0-360) :";I:IFI<0ORI>360THEN110
 115 I=I*255/360:POKE7122,I
 120 INPUT"UNTERE GRENZE:";SA
 130 INPUT" OBERE GRENZE:";SE
 140 DT=(SE-SA)/Q
 190 X0=9E9:Y0=9E9:Z0=9E9
 194 X9=-9E9:Y9=-9E9:Z9=-9E9
 200 FORI=0TOQ:T=SA+I*DT
 210 GOSUB5:Y=-Y:X(I)=X:Y(I)=Y:Z(I)=Z
 220 IFX>X9THENX9=X
 230 IFY>Y9THENY9=Y
 240 IFZ>Z9THENZ9=Z
 250 IFX<X0THENX0=X
 260 IFY<Y0THENY0=Y
 270 IFZ<Z0THENZ0=Z
 280 NEXT:XM=(X9+X0)/2:YM=(Y9+Y0)/2:ZM=(Z9+Z0)/2
 300 FORI=0TOQ
 310 X(I)=X(I)-XM
 320 Y(I)=Y(I)-YM
 330 Z(I)=Z(I)-ZM
 340 D=X(I)*X(I)+Y(I)*Y(I)+Z(I)*Z(I):IFD>D9THEND9=D
 390 NEXT:D9=SQR(D9):C=127/D9
 400 FORI=0TOQ
 410 X=X(I)*C:IFX<0THENX=256+X
 420 Y=Y(I)*C:IFY<0THENY=256+Y
 430 Z=Z(I)*C:IFZ<0THENZ=256+Z
 440 POKE6843+I,X:POKE6587+I,Y:POKE6331+I,Z
 450 NEXT
 460 PRINT"■BERECHNUNG BEENDET.
 465 PRINT"BITTE DRUECKEN SIE DIE TASTEN:■"
 470 PRINT"'1' BZW.'3':FUER DREHUNG UM DIE Z-ACHSE■"
 480 PRINT"',' BZW.'5':FUER AENDERUNG DES ELEVATION-WINKELS"
 490 PRINT"■■UNTERBRECHUNG DER LAUFENDEN ANZEIGE"
 495 PRINT"DURCH ZUSAETZLICHES DRUECKEN DER 'SHIFT'-TASTE"
 500 POKE7132,0:POKE7131,0:SYS8154:CLR:POKE525,0:GOTO100
READY.
```

(bitte umblättern)

Dreidimensionale Darstellung auf dem CBM 2001

```
 4 DATA216,24,173,224,3,141,185,3,105,176,176,11,24,173,225,3,141,186,3
 8 DATA105,206,144,1,96,78,185,3,169,0,170,105,1,78,186,3,144,2,10
 12 DATA10,168,173,186,3,10,10,109,186,3,10,10,144,2,232,232,10,144,1
 16 DATA232,24,109,185,3,141,136,3,141,185,3,138,105,128,141,137,3,141,186
 20 DATA3,173,0,129,162,15,221,230,3,240,10,202,16,248,173,227,3,240,34
 24 DATA162,0,138,141,136,3,152,77,226,3,170,173,226,3,240,7,138,45,136
 28 DATA3,76,180,3,138,13,136,3,170,189,230,3,141,65,129,24,173,224,3
 32 DATA109,228,3,48,4,201,80,48,2,169,0,141,224,3,24,173,225,3,109
 36 DATA229,3,48,4,201,50,48,2,169,0,141,225,3,96,2,16,0,0,255
 40 DATA0,32,126,124,226,123,97,255,236,108,127,225,251,98,252,254,160
 104 DATA0,0,6,9,12,16,19,22,25,47,31,34,37,40,43,46,49,51,54,57,60,63
 108 DATA65,68,71,73,76,78,81,83,85,88,90,92,94,96,98,100,102,104,106,107,10
 112 DATA111,112,113,115,116,117,118,120,121,122,122,123,124,125,125,126,126
 116 DATA126,127,127,127,127,127,127,127,126,126,126,125,125,124,123,122,122
 120 DATA121,120,118,117,116,115,113,112,111,109,107,106,104,102,100,98,96,9
 124 DATA92,90,88,85,83,81,78,76,73,71,68,65,63,60,57,54,51,49,46
 128 DATA43,40,37,34,31,28,25,22,19,16,12,9,6,3,162,0,142,209,27
 132 DATA189,187,23,141,206,27,189,187,22,141,207,27,189,188,23,141,196,27,1
 136 DATA188,22,141,197,27,32,144,28,174,209,27,236,208,27,240,6,232,142,209
 140 DATA27,208,215,96,173,206,27,141,224,3,173,207,27,141,225,3,162,0,56
 144 DATA189,196,27,253,206,27,157,198,27,169,0,141,187,27,141,189,27,157,20
 148 DATA27,157,202,27,42,42,105,255,157,228,3,157,204,27,16,11,24,189,198
 152 DATA27,73,255,105,1,157,198,27,232,224,1,240,204,173,198,27,205,199,27
 156 DATA240,111,162,0,144,1,232,189,198,27,141,188,27,138,73,1,168,185,198
 160 DATA27,141,190,27,32,90,29,173,191,27,157,202,27,169,0,157,228,3,185
 164 DATA196,27,217,224,3,208,3,76,49,29,32,49,29,189,204,27,48,17,24
 168 DATA189,202,27,125,200,27,157,200,27,144,225,254,206,27,176,220,189,200
 172 DATA253,202,27,157,200,27,176,209,222,206,27,144,204,152,72,138,72,189,
 176 DATA27,10,189,206,27,105,0,157,224,3,32,58,3,104,170,104,168,96,173
 180 DATA196,27,205,224,3,208,3,76,58,3,32,58,3,56,176,239,169,0,141
 184 DATA191,27,141,192,27,141,193,27,141,195,27,169,1,141,194,27,173,187,27
 188 DATA208,5,173,188,27,240,15,173,190,27,208,11,173,189,27,208,6,169,255
 192 DATA141,195,27,96,173,188,27,205,190,27,144,95,208,8,173,187,27,205,189
 196 DATA27,144,85,173,188,27,74,8,205,190,27,144,29,208,11,40,173,187,27
 200 DATA106,205,189,27,144,18,8,40,14,189,27,46,190,27,14,193,27,46,194
 204 DATA27,56,176,199,40,173,192,27,13,194,27,141,192,27,173,191,27,13,193
 208 DATA27,141,191,27,56,173,187,27,237,189,27,141,187,27,173,188,27,237,190
 212 DATA27,141,188,27,208,6,173,187,27,208,1,96,173,193,27,201,1,240,243
 216 DATA78,194,27,110,193,27,78,190,27,110,189,27,56,176,131,173,3,2,201
 220 DATA255,240,249,201,2,240,22,201,34,240,28,201,25,240,35,201,26,208,233
 224 DATA173,220,27,237,210,27,141,220,27,96,173,219,27,237,210,27,141,219,27
 228 DATA96,173,219,27,24,109,210,27,141,219,27,96,173,220,27,24,109,210,27
 232 DATA141,220,27,96,138,72,73,64,41,127,170,189,225,27,170,104,41,192,240
 236 DATA23,201,192,240,19,138,73,255,24,105,1,96,138,72,41,127,170,189,225
 240 DATA27,170,104,48,237,138,96,160,0,174,220,27,32,96,30,141,224,27,174

READY.
```


S & B SOFTWARE bietet an:

Qualitätssoftware zu vernünftigen Preisen – für CBM 2001/3032/8032

Zum Beispiel:
Schach 2.6 für 3032 45,– DM
Textverarbeitung ab 168,– DM
Dateiprogramme ab 53,– DM

Über 300 interessante Programme für Business, Schulen und Hobby.

NEU NEU NEU NEU NEU NEU

NEWTIM S das unentbehrliche Werkzeug für den professionellen Programmierer. Er erweitert den Befehlsvorrat des CBM um ca. 45 Befehle und enthält ein komplettes Floppy-DOS. Der NEWTIM S ist in einem 4K-E-PROM untergebracht und besetzt dadurch keinen RAM-Platz (nur für CBM 3032 und CBM 8032).

Weitere E-PROMS wie SUPERDISCBASIC, DOS und andere Programmierhilfen auf Anfrage.

Vollständiger Software-Katalog gegen Schutzgebühr von 8,– DM in Briefmarken.

S & B SOFTWARE GmbH

Postfach 369 · **5414 Vallendar** · Telefon (02 61) 3 59 19

TRS-80-Programmlösungen

Individuelle Programmentwicklung: Das ist unsere Spezialität. Ihren Bedürfnissen entsprechend, entwickeln wir jedes kommerzielle oder technisch/wissenschaftliche Programm (für alle TRS-80-Modelle). Vorteil: „maßgeschneidertes" Programm.

Standardprogramme: Wählen Sie aus unserer Programmbibliothek. Z.B.: Lohn/Gehalt; Adressenverwaltung; Lagerhaltung; Fakturierung allgemein; Fakturierung für Dentallaboratorien. Vorteil: Preisgünstig.

Für Hobbyisten: Spiel- und Dienstprogramme von Microsoft (das Softwarehaus, das BASIC für den TRS-80 und andere Microcomputer entwickelt hat). Z.B.: BASIC-Compiler; Olympischer Zehnkampf; Adventure; Editor Assembler plus.

Fordern Sie unverbindlich unsere Liste T81/2 an!

COLMAN-COMPUTER-SERVICE

Dipl.-Mathematiker Richard Colman

Postfach 14 31 61, 4300 Essen 14, Telefon (02 01) 51 04 67.

```
144 DATA219,27,32,96,30,141,223,27,174,220,27,32,69,30,141,222,27,174,219
248 DATA27,32,69,30,141,221,27,174,224,27,185,187,25,32,107,31,173,217,27
252 DATA141,213,27,173,218,27,141,224,3,174,222,27,185,187,26,32,107,31,24
256 DATA173,217,27,109,213,27,141,213,27,173,218,27,109,224,3,141,224,3,174
260 DATA222,27,185,187,25,32,107,31,173,217,27,141,212,27,173,218,27,141,225
264 DATA3,174,224,27,185,187,26,32,107,31,56,173,212,27,237,217,27,141,212
268 DATA27,173,225,3,237,218,27,141,225,3,173,221,27,72,170,173,212,27,72
272 DATA173,225,3,32,107,31,173,218,27,141,225,3,173,217,27,141,212,27,104
276 DATA170,104,32,107,31,173,218,27,24,109,212,27,141,212,27,144,3,238,225
280 DATA3,174,223,27,185,187,24,32,107,31,173,217,27,24,109,212,27,141,212
284 DATA27,173,218,27,109,225,3,141,225,3,24,173,224,3,105,128,74,74,153
288 DATA187,23,24,173,225,3,105,128,74,74,74,153,187,22,74,24,121,187,22
292 DATA153,187,22,204,211,27,208,1,96,200,76,148,30,142,214,27,141,215,27
296 DATA77,214,27,72,173,215,27,16,5,73,255,24,105,1,42,141,215,27,173
300 DATA214,27,16,8,73,255,24,105,1,141,214,27,162,0,142,216,27,142,218
304 DATA27,142,217,27,232,138,45,214,27,240,19,24,173,215,27,109,217,27,141
308 DATA217,27,173,218,27,109,216,27,141,218,27,224,64,240,13,138,24,10,170
312 DATA46,215,27,46,216,27,76,156,31,104,16,16,173,217,27,73,255,141,217
316 DATA27,173,218,27,73,255,141,218,27,96,173,4,2,240,1,96,32,4,30
320 DATA32,110,30,169,53,141,17,232,32,54,226,174,211,27,202,142,208,27,32
324 DATA97,28,169,60,141,17,232,76,218,31
999 POKE135,22:POKE134,187          :REM/CBM: POKE 53,22: POKE 52,187
1000 FORI=826TO1013:READP:POKEI,P:NEXT
1010 FORI=7137TO8191:READP:POKEI,P:NEXT:END
1995 AD=7136:I=100
19910 I=I+4:Q=0:PRINT:PRINTI;"DATA";
19915 AD=AD+1:IFAD>8191THENSTOP
19920 D$=STR$(PEEK(AD))
19930 D$=RIGHT$(D$,LEN(D$)-1)
19935 IFQ<18THEND$=D$+","
19942 PRINTD$;:Q=Q+1:REM
19949 IFQ>18GOTO19910
19950 GOTO19915
READY.
```


AUF DIE SOFTWARE KOMMT ES AN!

System- und Anwendersoftware für CP/M-kompatible Mikrocomputersysteme
Betriebssysteme (CP/M, MP/M, ...)
Interpreter (BASIC, PASCAL, APL, ...)
Compiler (BASIC, FORTRAN, COBOL, PL/I, PASCAL, ...)
Hilfsprogramme (SORT, PLOT, Konvertierungen, ...)
Anwenderprogramme (FIBU, Textverarbeitung, ...) Immer mit Originallizenz. Jetzt auch in Sonderformaten (z.B. Superbrain, PSI-80, North-Star, Heath, ...).

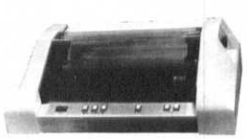
Fordern Sie unsere aktuelle Preisliste an, Sie werden (positiv) überrascht sein!

BSP Thomas Krug EDV-Beratung,
Neuprüll 7/Postfach 11 03 24
Telefon (09 41) 9 39 45

Soft- und Hardwarevertrieb
D-8400 Regensburg
Telex 06 52 510 krug d

P.S.: Wir liefern in der Regel ab Lager.

ANADEX* MATRIXDRUCKER



die Profis: DP-9000, DP-9001, DP-9500, DP-9501

- 80-220 Zeichen/Zeile, 50-200 Zeilen/Min.
- Groß-, Klein-, Spreitzschrift, Unterlängen
- Vollgraphic, Nadeln einzeln ansteuerbar
- Einschließlich aller Standard-schnittstellen, V-24, 20-60 mA, Centronics Parallel, umschaltbar
- Tabulierung, Formfeed, X-ON/X-OFF, STX/ETX/LRC, DEC-Protokoll, Barcodes u.v.a.m. programmierbar !

Einzelpreis: ab DM 4.463,50 (inkl. MWSt.)+Zustellung · OEM- und Händlerpreise auf Anfrage.
***Werksvertretung**

SCHWIND DATENTECHNIK G M B H

Ebenböckstr. 4, D-8000 München 60, Tel. (089) 8 34 97 16, Telex 05-213097

Heino Velder

Zeichnen mit CBM 2001

Mit Hilfe der Blockgrafik des CBM kann man eine Auflösung von 80 Punkten in X-Richtung und mit 50 Punkten in Y-Richtung erzielen. Mit dem Befehl „DRAW" und der anschließenden Angabe der Koordinaten kann man nun sehr einfach Plot-Programme gestalten, wobei der Koordinatensprung in der linken unteren Bildschirmecke liegt.

Beispiel:
```
DRAW 10,20
```

oder mit Variablen:
```
DRAW INT
(SIN(X/2))*12,ABS(3/-Y)
```

Das Programm selbst wird weitgehend durch das Assemblerlisting erläutert. Interessant ist jedoch der Teil, in dem die Koordinaten in das Maschinenprogramm übernommen werden.
Mit JSR $CCA4 wird ein dezimaler Ausdruck (dessen Adresse sich in $C9, CA befindet) geholt und anschließend mit JSR $D6D0 nach binär umgewandelt. Das Ergebnis befindet sich dann in den Speicherzellen $08, 09, wobei nun geprüft wird, ob der zulässige Wertebereich (X = 0—79, Y = 0—49) überschritten wurde. Mit JSR $CCA4 wird automatisch der Zeiger ($C9, CA) erhöht. Bevor jedoch die Y-Koordinate ausgelesen wird, muß der Zeiger um 1 erhöht werden, da dieser auf das Komma zeigt, was zu einer Fehlermeldung führen würde.

Wegen des geänderten Betriebssystems sind bei CBM vier Adressen zu ändern:

Die ZERO-PAGE Adressen:
$08, 09 → $11, 12
$C9, CA → $77, 78

und die Routinen:
$CCA4 → $CC8B
$D6D0 → $D6D2

Assemblerlisting zu 'DRAW'

```
1F3E:  20 A4 CC   JSR  $CCA4   ← holt X-Koordinate aus BASIC
1F41:  20 D0 D6   JSR  $D6D0   ← dezimal nach binär
1F44:  A5 09      LDA  $09     ┐
1F46:  D0 06      BNE  $1F4E   | Prüft ob X grösser als
1F48:  A5 08      LDA  $08     |
1F4A:  C9 50      CMP  =$50    | #79 ist
1F4C:  30 0C      BMI  $1F5A   ┘
1F4E:  A2 16      LDX  =$16    ┐
1F50:  BD E7 1F   LDA  $1FE7,X | Schreibt
1F53:  20 D2 FF   JSR  $FFD2   |
1F56:  CA         DEX          | '?ILLEGAL POINT ERROR'
1F57:  D0 F7      BNE  $1F50   | (Routine beenden)
1F59:  60         RTS          ┘
1F5A:  48         PHA          ← X-Koordinate auf Stack
1F5B:  E6 C9      INC  $C9     ┐
1F5D:  D0 02      BNE  $1F61   | Pointer erhöhen
1F5F:  E6 CA      INC  $CA     ┘ (zeigt sonst auf das Komma)
1F61:  20 A4 CC   JSR  $CCA4   ← holt Y-Koordinate aus BASIC
1F64:  20 D0 D6   JSR  $D6D0   ← umwandeln (Ergebnis in $08,09)
1F67:  68         PLA          :
1F68:  AA         TAX          ← X-Reg. = X-Koordinate
1F69:  A5 09      LDA  $09     ┐
1F6B:  D0 E1      BNE  $1F4E   | Prüft ob Y grösser als
1F6D:  A5 08      LDA  $08     | #49, dann Fehlermeldung
1F6F:  C9 32      CMP  =$32    |
1F71:  10 DB      BPL  $1F4E   ┘
1F73:  A0 00      LDY  =$00    ┐
1F75:  4A         LSR          | Y-Koord. dividiert durch 2
1F76:  90 02      BCC  $1F7A   |
1F78:  A0 02      LDY  =$02    | $00 enthalt Feinabstufung
1F7A:  84 00      STY  $00     ┘
1F7C:  A8         TAY          ┐
1F7D:  A9 83      LDA  =$83    |
1F7F:  85 02      STA  $02     | Indirekte Adresse $01, 02
1F81:  A9 E8      LDA  =$E8    |
1F83:  38         SEC          | wird auf Zeilenanfang, also
1F84:  E9 28      SBC  =$28    |
1F86:  B0 02      BCS  $1F8A   | Y-Richtung, gesetzt
1F88:  C6 02      DEC  $02     |
1F8A:  88         DEY          |
1F8B:  10 F6      BPL  $1F83   |
1F8D:  85 01      STA  $01     ┘
1F8F:  8A         TXA          ┐
1F90:  A6 00      LDX  $00     | X-Koord. dividiert durch 2
1F92:  4A         LSR          |
1F93:  90 01      BCC  $1F96   | Y-Register = X-Richtung (grob)
1F95:  E8         INX          |
1F96:  A8         TAY          |
1F97:  A9 F8      LDA  =$F8    ┐
1F99:  18         CLC          | $00 auf Tabelle einstellen,
1F9A:  69 08      ADC  =$08    | je nachdem welcher der vier
1F9C:  CA         DEX          | Punkte in der Matrix ange-
1F9D:  10 FA      BPL  $1F99   | sprochen wird
1F9F:  85 00      STA  $00     ┘
1FA1:  B1 01      LDA  ($01),Y ← Zeichen vom Bildschirm holen
1FA3:  48         PHA          :
1FA4:  29 7F      AND  =$7F    ← Bit 7 (REVERSE) löschen
1FA6:  A2 07      LDX  =$07    ┐
1FA8:  DD C0 1F   CMP  $1FC0,X | ist Zeichen ein Graphiksymbol?
1FAB:  F0 07      BEQ  $1FB4   | X-Register zeigt dann in die
1FAD:  CA         DEX          | Tabelle
1FAE:  10 F8      BPL  $1FA8   ┘
1FB0:  A9 20      LDA  =$20    : nein, dann ACCU = SPACE
1FB2:  A2 07      LDX  =$07    : (Text wird dann überschrieben)
1FB4:  8A         TXA          ┐
1FB5:  18         CLC          | Feinabstufung zum X-Register
1FB6:  65 00      ADC  $00     |
1FB8:  AA         TAX          ┘
1FB9:  68         PLA          :
1FBA:  5D C8 1F   EOR  $1FC8,X ← Graphikpunkt invertieren,
1FBD:  91 01      STA  ($01),Y ← und zurück in den Bildschirm
1FBF:  60         RTS          : 'DRAW' ** Ende **
1FC0:                          :
1FC0:                          :
1FC0:                          :
1FC0:  7B 6C 7E 7C 61 62 7F 20 *
1FC8:  5B 0E 1F 83 1F 0E 83 5B *
1FD0:  19 4C 01 9D 9D 19 01 4C *  Daten
1FD8:  1A 13 5E 9E 1A 9E 13 5E *
1FE0:  84 8D 9C 5C 8D 9C 84 5C *
1FE8:  8D 52 4F 52 52 45 20 54 *┐
1FF0:  4E 49 4F 50 20 4C 41 47 *| Text für die Fehlermeldung
1FF8:  45 4C 4C 49 3F 0D 00 00 *┘
```

Da das Programm im ungeschützten Bereich liegt, muß er zuvor mit POKE 134,60/POKE/135/30 gesichert werden.

Dieses Programm kann ebenfalls leicht in einen anderen Speicherbereich verlegt werden. Es müssen lediglich die Zeiger auf der Tabelle geändert werden. Natürlich ist auch die Adresse in der Befehlstabelle entsprechend zu ändern.

Bei dieser Ausführung sind die folgenden Daten einzugeben:

```
$03D8 . 44 52 41 57 00 00 3E 1F
```

und Sie haben Ihren BASIC-Befehlssatz mit „DRAW" erweitert.

```
Beispiel zu 'DRAW'

 5 SYS 826
 10 FORX=0TO79
 20 W=W+0.0976
 30 Y=INT(25-SIN(W)*25)
 40 DRAW X,Y:NEXT
 50 SYS 837
READY.
```

```
Beispiel zu 'WHITE

 10 FORA=0TO100:NEXT:WHITE:GOTO 10
READY.
```

```
Beispiel zu 'COPY'

 10 COPY
 20 FORA=0TO5
 30 PRINT"ALLLE AUSGABEN GEHEN AN DEN DRUCKER"
 40 NEXT
 50 COPY
READY.
```


Pro-bleme?

Wir, als Systemhaus, sind darauf spezialisiert, Ihre Probleme zu lösen. Ein Beispiel aus unserem umfangreichen Angebot:

Lagerbuchhaltung
Textverarbeitung
Statistik-System
Finanzbuchhaltung
Kalkulation

Leisten Sie sich die Qual der Wahl, denn die Auswahl Ihres Systems, mit der dazugehörigen Software, ist keine Sache per Postkarte. Lassen Sie sich von Fachleuten beraten.

Auch mit Ihrem eventuell auftretenden Serviceproblemen sind Sie bei uns in besten Händen. Eine komplett ausgerüstete Werkstatt mit geschulten Ingenieuren bringt Ihren Rechner schnell wieder in Ordnung!

Natürlich liefern wir auch andere Software: Hilfsprogramme und Spiele sind ebenso in unserem reichhaltigen Softwareprogramm vertreten wie Businessprogramme.

Qualität muß nicht teuer sein!

Fordern Sie Vorabinformationen an oder kommen Sie doch einfach mal vorbei!!

CompuSoftpak

6360 Friedberg 1, Usavorstadt 5
Telefon 0 60 31/72 11

L. Sprünken

Funktionen auf TRS 80

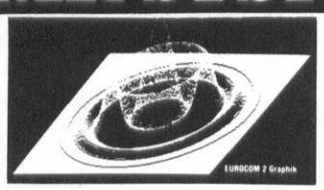
Das folgende Programm zeichnet mathematische Funktionen und deren Ableitungen (wegen mangelnder Genauigkeit max. bis zur 4. Diff.) mit einer Auflösung von 128 × 42 Punkten + 2 zusätzlichen „Informationszeilen" am unteren Bildschirmrand für Intervall, Maximum + Minimum, „Betriebsart" und Zähler für die Rechenkontrolle. Das Programm berechnet selbst den richtigen Maßstab und die Lage des verschiebbaren Koordinatensystems.

```
80 CLEAR200
100 ON ERROR GOTO 1680
120 '>>> SPRUENKEN - SOFTWARE   (C) 1979 <<<
140 N0=0:CLS:DIMA(141),A0(141):GOTO880
200 PRINT:PRINT"EINGABE DER INTERVALLGRENZEN :"
220 INPUT"LINKS";A
240 INPUT"RECHTS";B
260 IFB>ATHEN300
280 PRINT"BITTE MATHEMATISCH RICHTIG!":GOTO220
300 CLS
320 PRINT@954,"COUNT";
330 PRINT@940,"DARST.";:PRINT@1004,"-FKT";
340 PRINT@896,"INTERVALL";:PRINT@960,A;"...";B;
360 PRINT@917,"MIN.","MAX.";
400 V1=-1.7E38:V2=-V1:X1=ABS((B-A)/128)
420 R0%=-126*SGN(A)
440 FORR%=0TO140
460 X0=X:V0=V:X=A+R%*X1
500 IFSGN(X)-SGN(X0)=2 ORX=0 THEN R0%=R%
520 PRINT@1018,R%;
540 GOSUB1260
560 A(R%)=V
570 IFR%>128THEN620
580 IFV>=V1THENV1=V
600 IFV<=V2THENV2=V
620 NEXT
640 PRINT@977,V2;:PRINT@990,V1;
660 H=(ABS(V1)+ABS(V2))/40:H0=ABS(V1/H)
680 FORP%=0TO60:PRINT@INT(H0/3)*64+P%,".";:NEXTP%:PRINT"X->";
700 IF ABS(SGN(A)+SGN(B))<2 THENFORP%=0TO13:PRINT@INT(R0%/2)+P%*64,".";:NEXT:
    PRINT@INT(R0%/2),CHR$(91);:PRINT@INT(R0%/2)+64,"V";
720 FORR%=0 TO 126
730 IFT6%<>1 THEN 760
740 FORT7%=H0 TO H0-INT(A(R%)/H) STEP (-1)*SGN(A(R%));SET(R%,T7%):NEXTT7%:
    GOTO800
760 SET(R%,H0-INT(A(R%)/H))
800 NEXT
820 PRINT@954,"READY?";
840 PRINT@1018,"<INP>";
850 T6%=0
860 IF INKEY$="" THEN 1080
880 CLS:PRINT">>> PLOTTER-PROGRAMM ZUM ZEICHNEN MATHEMATISCHER FUNKTIONEN <<<"
900 PRINT:PRINT"DRUECKEN SIE:   ";CHR$(92)
920 R9%=0
940 PRINT:PRINT". . . . . **1**   ZUM PROGRAMM LOESCHEN *"
960 PRINT  ". . . . . **2**   FUER NEUE FUNKTIONSEINGABE"
980 PRINT  ". . . . . **3**   FUER NEUEN WERTEBEREICH"
1000 PRINT". . . . . **4**   ZUM ZEICHNEN DER ABLEITUNG"
1020 PRINT". . . . . **5**   ZUM AUFLISTEN DER FUNKTION *"
1040 PRINT    ". . . . . **6**   ZEICHNEN DER AUSGEFUELLTEN FUNKTION"
1050 PRINT    ". . . . . **7**   ZEICHNEN DER AUSGEFUELLTEN ABLEITUNG"
1060 PRINT:PRINT:PRINT:PRINT">>>>>   A U F   I H R E M   V I D E O - T E R M I N
     A L   <<<<<"
1080 ON VAL(INKEY$) GOTO 1120,1140,1180,1300,1200,1220,1230
1100 GOTO1080
1120 NEW
1140 CLS
1160 PRINT"AENDERN SIE IHRE FUNKTION FOLGENDERMASSEN AB,":PRINT"INDEM SIE MIT
     H V = . . . .        BEGINNEN":PRINT:PRINT"STARTEN SIE DANN ERNEUT MIT <RUN>
":PRINT:PRINTCHR$(27);:PRINTCHR$(27);CHR$(29);:EDIT1260
1180 CLS:PRINT:PRINT:PRINT:N0=0:PRINT:GOTO200
1200 CLS:PRINT"DAS PROGRAMM BEINHALTET FOLGENDE FUNKTION:":LIST1260
```

```
1220 T6%=1:GOTO1180
1230 T6%=1:GOTO1300
1240 ' >>>>>>>> FUNCTION F(X) <<<<<<<<
1260 Y=SIN(X)
1280 RETURN
1300 ' >>>>>>>> DIFF F'(X) <<<<<<<<
1310 N0=N0+1
1320 CLS:PRINT@954,"COUNT";:PRINT@917,"MIN.","MAX";:PRINT@896,"INTERVALL";:PRINT
     @960,A;"...";B:PRINT@940,"DARST.";:PRINT@1003,N0;".DIF";
1340 Z2=1.7E38:Z1=-Z2:Z1=0:Z2=0:FOR R%=0 TO 140-N0
1360 PRINT@1018,R%;
1380 A0(R%)=(A(R%+1)-A(R%))/X1
1390 IFR%>128THEN1430
1400 IFA0(R%)>=Z1 THEN Z1=A0(R%)
1420 IFA0(R%)<=Z2 THEN Z2=A0(R%)
1430 A(R%)=A0(R%)
1440 NEXT R%
1460 PRINT@977,Z2;:PRINT@990,Z1;
1480 I=ABS(Z1-Z2)/40:I0=Z1/I
1500 FORP%=0TO60:PRINT@INT(I0/3)*64+P%,".";:NEXTP%:PRINT"X->";
1520 IFABS( SGN(A)+SGN(B))<2 THENFORP%=0TO13:PRINT@INT(R0%/2)+P%*64,".";:NEXT:
     PRINT@INT(R0%/2),CHR$(91):PRINT@INT(R0%/2)+64,"A";
1540 FORR%=0 TO 126
1550 IFT6%<>1THEN1560 ELSE FORT7%=I0 TO I0-INT(A0(R%)/I)STEP(-1)* SGN(A0(R%)):
     SET(R%,T7%):NEXTT7%:GOTO1580
1560 SET(R%,I0-INT(A0(R%)/I))
1580 NEXT
1600 GOTO820
1680 IF ERR=20 THEN X=X+(B-A)/1000 : RESUME  ELSE CLS: PRINT"FEHLER IN IHRER
     FUNKTION!":RESUME 1160
1800 END
```


MK-SYSTEMTECHNIK

Pater-Meyer-Str. 6 · Postfach 730 · 6728 Germersheim/Rhein
Tel. (0 72 74) 27 56 · Telex 4 53 500
7500 Karlsruhe, Kriegstr. 164 · Tel. (07 21) 2 92 43



EUROCOM-1 449,—
6802 CPU, 1,25 RAM, 2k Betriebssystem, Hex 1/0, 40 E/A-Leitungen. Beschreibung in Chip 11/79.

EUROCOM-2 1670,—
Einplatinencomputer im Doppel-Europaformat. Sehr leistungsf. Graphik.
- 48k RAM, 4k Betriebssystem
- 6809 CPU (interne 16 Bit Struktur!)
- 128 Zeichen, Groß/Kleinschreibung
- Graphik und Text beliebig mischbar
- Kansas-City-Standard-Interface
- 40 E/A-Leitungen, RS 232 C-Anschluß
- Ausbaubar auf Farbgraphik

Zubehör für EUROCOM-2

Floppy-Controller Single-Density ohne DMA	1127,—
Floppy-Controller Double-Density mit DMA	1692,—
5"-Laufwerk	847,—
BUS-Karte	84,—
RAM-Karte 32k	779,—
RAM-Karte 96k	1977,—
Parallel Ein/Ausgabe VIA	562,—
Serielle Ein/Ausgabe ACIA	562,—
5A EUROCOM II-Netzteil	384,—
ASCII-Tastatur	279,—
Joystick	110,—

Software f. EUROCOM-2

BASIC	220,—
Assembler/Editor	220,—
DEBUG	179,—
Disassembler	113,—

wahlweise auf Audiocassette oder Mini-Digitalcassette.

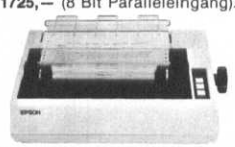
DAI Computer 2860,—
- 24k ROM BASIC, halbkompilierend
- Farbgraphik (12 Betriebsarten) mit bis zu 255 X 335 Punkten und Farben.
- 3 Tongeneratoren; 1 Rauschgenerator (Anschluß an eine Stereoanlage)

48k RAM-Speicher 2860,—
Floppy mit Controller 2095,—
Arithmetikprozessor 798,—

12"-Monitor 748,—
SANYO-12-Zoll-Monitor (ca. 31 cm) für höchste Ansprüche. 18 MHz-Bandbreite. BAS-Eingang. Dieses Gerät paßt, auch optisch, ideal um Apple II und ist als Monitor für augenschonende Dauerarbeit geradezu unerläßlich. Auch für Graphik-Darstellungen, z.B. mit APPLE oder EUROCOM II bietet sich dieser Monitor an.

BMC 12"-Monitor grün 555,—

MX 80 F/T ab 1725,—
m. einem Interface n. Wahl 1995,—
Der neue MX 80 F/T kann entweder randgelochtes Endlospapier oder Einzelblätter bedrucken.— Interfacekarten für alle gängigen Rechnersysteme sind lieferbar: PET/CBM, TRS 80, MZ 80 K, IEEE 488 (HP) oder RS 232 C (V24) — Interface z. B. für HEATH-Computer, NASCOM oder COMPUCORP.— Sie erhalten den MX 80 F/T auch ohne gesondertes Interface für DM 1725,— (8 Bit Paralleleingang).



WATANABE Plotter
DIN A 3-Format 3230,—

An alle Mikrocomputer mit Parallelschnittstelle anschließbar.

NASCOM-1 775,—

Fertiggerät	895,—
Netzteil Kit ± 5 V, ± 12 V, 5 A	210,—

NASCOM-2 1785,—
System mit 8k RAM, BASIC, 2k Monitor 1785,—
System mit 16k RAM, BASIC, 2k Monitor 1975,—

FLOPPY DISK 1749,—
5-Zoll-Floppy, Fertiggerät mit DOS, BASIC, Macroassembler, Debug, Texteditor; für NASCOM-1 oder NASCOM-2 — 1 Jahr Software-Pflege kostenlos.

5-Zoll-Floppy (s.o.) mit Gehäuse und Netzteil für zwei Laufwerke 2144,—

NAS-SYS-Assembler 4k	299,—
NAS-SYS-Disassembler	199,—
NAS-SYS-Debug	75,—

Schach für NASCOM 1/2 128,—
Schachgraphik ROM für NASCOM-2 98,—
Graphik-Zusatzkarte f. NASCOM-1 198,—

NASCOM-JOURNAL
Zeitschrift für den ernsthaften NASCOM-Anwender. Zahlreiche Programmbeispiele, Anregungen zur Hardware etc.—

Jahresabonnement	48,—
Jahrgang 1980 komplett	39,—

VIDEO-GENIE 1395,—
System (neue Ausführung)

Expansion Interface mit 32k Speichererweiterung 1645,—
Floppylaufwerk mit Gehäuse und Netzteil, 40-Track-Ausführung 995,—
Zweites Laufwerk 40 Track 775,—
Verbindungskabel für 2 Laufwerke 78,—

apple® computer

APPLE II EURO PLUS
Einzelne Geräte bitte anfragen.

64k APPLE II EUROPLUS 3350,—
Ideal für PASCAL, FORTRAN

SYSTEM B 4998,—
APPLE II EUROPLUS 48k Speicher mit 1 Floppy Disk-Laufwerk sowie 12-Zoll-Monitor grün

SYSTEM C 6846,—
APPLE II EUROPLUS 48k Speicher mit Floppy Disk-Laufwerk, MX 80 Matrixdrucker mit APPLE Interface und 12-Zoll-Monitor grün

1. Floppy-Laufwerk, DOS 3.3	1538,—
2. Floppy-Laufwerk ohne Controller	1198,—
PASCAL Language System Version 1.1 (neu)	1198,—
Kalender/Uhr-Karte	355,—
Serienschnittstelle 50-9600 Bd.	455,—
Analog/Digitalwandler 3 3/4-stellig	318,—
Z 80 SOFTCARD mit CP/M und BASIC	895,—
Arithmetikprozessor	999,—
Graphics Tablet	2150,—
DOS 3.3 SET, erweitert von DOS 3.2.1 auf DOS 3.3	175,—

Software

Textverarbeitung	244,—
VISICALC	490,—
KASSIERER (Lagerhaltung/Fakturierung)	1099,—
Finanzbuchhaltung FIPAS	3495,—

BOOKSHOP
führt zahlreiche Bücher für Einsteiger, Profis, BASIC-Spezialisten, Maschinensprachprogrammierer u.v.a.m. Sie finden zahlreiche Dokumentationen zu Computern, Lehrbücher, Nachschlagewerke und andere nützliche Literatur.— Fordern Sie unseren kostenlosen BOOKSHOP-Katalog an.

®Eingetragenes Warenzeichen APPLE Computer Inc., Cupertino

Alle Preise schließen die Mehrwertsteuer ein. Versand per Nachnahme oder nach Vorausrechnung. Preisänderung, Preisirrtum und Zwischenverkauf vorbehalten

Aus der Trickkiste von „Computer-Profis": Tips und Tricks, die das Arbeiten am Computer erleichtern und die Computerei noch interessanter machen.
Hier unsere Auswahl:
Kalender für TI-59/PC 100A,
CBM 2001 wird vielseitiger,
Blick in die CPU-Register des CBM 2001,
CBM-Maschinencode im String gespeichert,
Hexadezimal-Tastatur mit TI-58,
Programmverschiebung mit dem TI-59,
Single Step für Basic-Programme auf dem TRS 80 Level II,
Basic-Load für CBM.

Linus Schneider

Kalender für TI-59/PC 100 A

Beschreibung

Das Programm berechnet und druckt den Kalender für jede Zeitdauer seit 1582, dem Datum der Einführung des Gregorianischen Kalenders. Auf Wunsch wird zusätzlich ein Titel ausgedruckt.
Pgm 20 D aus dem Standardmodul bestimmt für den ersten Tag des ersten Monats den Wochentag, der in Form einer Kennziffer (KZ.) erscheint:

0 = Samstag
1 = Sonntag
2 = Montag
3 = Dienstag
4 = Mittwoch
5 = Donnerstag
6 = Freitag

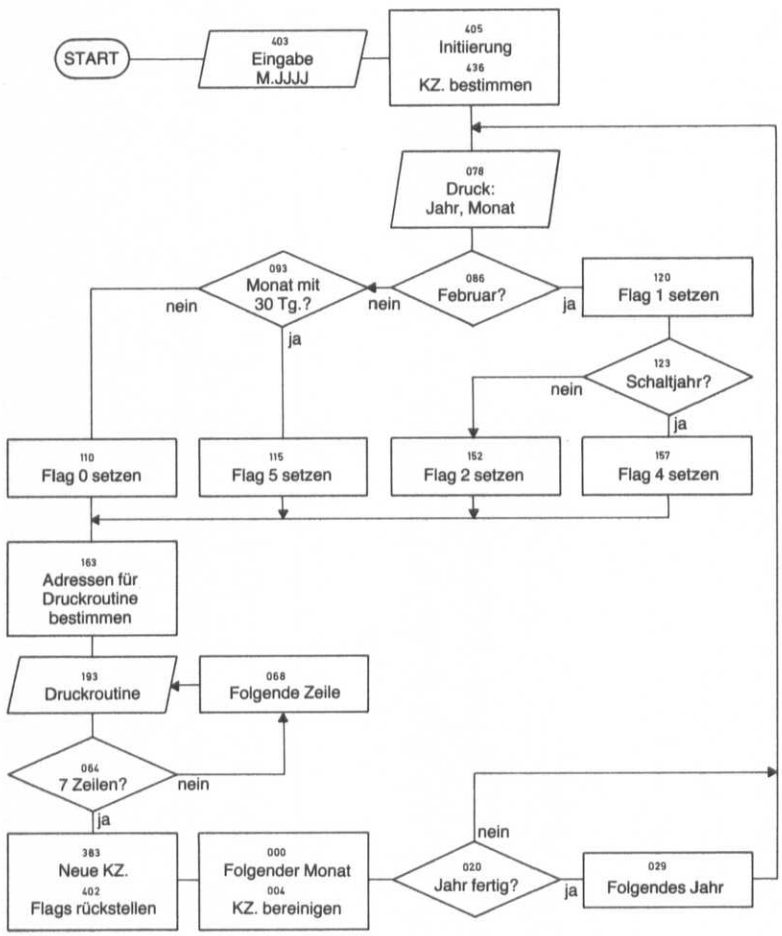
Die Kennziffer des ersten der Folgemonate wird nach der Länge des Vormonats bestimmt.
Für den Druck sind 13 verschiedene Zeilen notwendig. Jeder Monatsblock umfaßt davon 7 aufeinanderfolgende (Tabelle 1):
Die Kennziffer steuert den Eintritt in die Druckroutine. Wenn beispielsweise der Erste des Monats auf einen Sonntag fällt (KZ. 1), startet die Druckroutine bei Zeile Nr. 7 (Programmzeile 280) und druckt die Zeilen Nr. 7 bis 13 aus; ist aber der Erste ein Mittwoch (KZ. 4), werden die Zeilen Nr. 4 bis 10 ausgedruckt (grau getöntes Feld).
Um alle Zeilenviertel zu speichern, bräuchte es mindestens 42 Datenspeicher. Mit den 12 Monats- und den 7 Tagesnamen ergäben sich schon 61 Speicher. So viele stehen nicht zur Verfügung. Deshalb werden einzelne Zeilenviertel in Unterprogrammen aufgebaut, nämlich alle Zeilen von Op 01 in den Unterprogrammen C und B und die Zeilen Nr. 2 bis 8 von Op 02 im Unterprogramm A. Im Unterprogramm B werden auch noch die Druckzeilen gezählt und der Druckbefehl (Op 05) erteilt.
Ein weiteres Problem bieten die Monatslängen: April, Juni, September und November haben 30 Tage, die übrigen — mit Ausnahme des Februars — 31 Tage. Der Februar hat 28 Tage, in den Schaltjahren aber 29. Die Jahreszahlen der Schaltjahre sind Elemente der Menge

$$(V_4 \setminus V_{100}) \cup V_{400}$$

V_4 = {4, 8, 12, 16,...};
V_{100} = {100, 200, 300,...}
V_{400} = {400, 800, 1200,...}

Tabelle 1:

Zeile Nr.	Op 01	Op 02		Op 03		Op 04		Progr.-Zeile	Start bei KZ.
1		2		9	16	23	30	193	0
2		3	1	0	17	24	31	214	6
3		4	1	1	18	25		232	5
4		5	1	2	19	26		244	4
5		6	1	3	20	27		256	3
6		7	1	4	21	28		268	2
7	1	8	1	5	22	29		280	1
8	2	9	1	6	23	30		298	
9	3	10	1	7	24	31		317	
10	4	11	1	8	25			337	
11	5	12	1	9	26			350	
12	6	13	2	0	27			361	
13	7	14	2	1	28			372	



Die Flags 0, 1 und 2 steuern beim Druck die Länge der Monate, die übrigen Flags dienen dazu, aus der Monatslänge die neue Kennziffer zu bestimmen.
Mit den Unterprogrammen und den Flags wird die Druckroutine ziemlich unübersichtlich, was sich aber kaum vermeiden läßt. Aus dem Flußdiagramm ist ersichtlich, daß das Programm in einer endlosen Schleife läuft. Mit R/S kann es jederzeit unterbrochen werden.

Speicherbelegung

00: MMTT.JJJJ für Pgm 20
01: Pgm 20/Adressen für Wochentage
02: Pgm 20/belegt für Op 01
03: Pgm 20/Kennzahl
04: Adressen für Monatsnamen
05: Ordnungszahl der Monate
06: Jahrzahl
07: Kennzahl/Kontrolle Anzahl Druckzeilen
08 bis 59: belegt für Drucktext

(bitte umblättern)

Bedienung Speicherbereichs-Verteilung: 479.59 (6 Op 17)

Nr.		Eingabe
1	Programm und Speicher 08 bis 59 laden (2 Magnetkarten)	
2	Wenn Titel gewünscht wird	D
3	Ordnungszahl M des Anfangsmonats und Anfangsjahr JJJJ durch Dezimalpunkt getrennt eingeben (z.B. April 1941 = 4.1941)	M.JJJJ
4	Druckprogramm starten (die Druckdauer für 12 Monate beträgt etwa 5 min)	E
5	Programm beendigen	R/S
6	Neues Programm mit RST und Schritt 2 oder Schritt 3 beginnen	

Kalender für TI-59/PC 100A

```
000  69 □P     072  01  01     144  02  2      216  48  48     288  69 □P     360  12  B
001  25  25    073  42 STO     145  05  5      217  87 IFF     289  04  04     361  13  C
002  69 □P     074  02  02     146  95  =      218  00  00     290  43 RCL     362  43 RCL
003  24  24    075  69 □P     147  22 INV     219  02  02     291  36  36     363  26  26
004  43 RCL    076  21  21     148  59 INT     220  23  23     292  69 □P     364  69 □P
005  03  03    077  92 RTN     149  67  EQ     221  43 RCL     293  03  03     365  03  03
006  55  ÷     078  73 RC*     150  01  01     222  47  47     294  09  9      366  43 RCL
007  07  7     079  04  04     151  57  57     223  69 □P     295  11  A      367  25  25
008  95  =     080  69 □P     152  86 STF     224  04  04     296  02  2      368  69 □P
009  22 INV     081  04  04     153  02  02     225  43 RCL     297  12  B      369  02  02
010  59 INT     082  43 RCL     154  61 GTO     226  46  46     298  13  C      370  07  7
011  65  ×     083  06  06     155  01  01     227  69 □P     299  25 CLR     371  12  B
012  07  7     084  69 □P     156  59  59     228  03  03     300  87 IFF     372  13  C
013  85  +     085  06  06     157  86 STF     229  04  4      301  01  01     373  43 RCL
014  93  .     086  43 RCL     158  04  04     230  11  A      302  03  03     374  24  24
015  01  1     087  05  05     159  05  5      231  12  B      303  06  06     375  69 □P
016  95  =     088  32 X:T     160  03  3      232  13  C      304  43 RCL     376  03  03
017  59 INT     089  02  2     161  42 STO     233  43 RCL     305  35  35     377  43 RCL
018  42 STO     090  67  EQ     162  01  01     234  45  45     306  69 □P     378  23  23
019  07  07    091  01  01     163  07  7      235  69 □P     307  04  04     379  69 □P
020  01  1     092  20  20     164  48 EXC     236  04  04     308  43 RCL     380  02  02
021  03  3     093  04  4      165  07  07     237  43 RCL     309  34  34     381  08  8
022  32 X:T     094  67  EQ     166  42 STO     238  44  44     310  69 □P     382  12  B
023  43 RCL    095  01  01     167  03  03     239  69 □P     311  03  03     383  87 IFF
024  05  05    096  15  15     168  32 X:T     240  03  03     312  01  1      384  05  05
025  22 INV     097  06  6      169  01  1      241  05  5      313  02  2      385  03  03
026  77  GE     098  67  EQ     170  67  EQ     242  11  A      314  11  A      386  98  98
027  00  00    099  01  01     171  02  02     243  12  B      315  03  3      387  87 IFF
028  78  78    100  15  15     172  80  80     244  13  C      316  12  B      388  04  04
029  01  1     101  09  9      173  02  2      245  43 RCL     317  13  C      389  04  04
030  42 STO     102  67  EQ     174  67  EQ     246  43  43     318  43 RCL     390  00  00
031  05  05    103  01  01     175  02  02     247  69 □P     319  33  33     391  22 INV
032  08  8     104  15  15     176  68  68     248  04  04     320  87 IFF     392  87 IFF
033  42 STO     105  01  1      177  03  3      249  43 RCL     321  00  00     393  00  00
034  04  04    106  01  1      178  67  EQ     250  42  42     322  03  03     394  04  04
035  69 □P     107  67  EQ     179  02  02     251  69 □P     323  25  25     395  02  02
036  26  26    108  01  01     180  56  56     252  03  03     324  25 CLR     396  69 □P
037  61 GTO     109  15  15     181  04  4      253  06  6      325  69 □F     397  23  23
038  00  00    110  86 STF     182  67  EQ     254  11  A      326  04  04     398  69 □P
039  78  78    111  00  00     183  02  02     255  12  B      327  43 RCL     399  23  23
040  76 LBL     112  61 GTO     184  44  44     256  13  C      328  32  32     400  69 □P
041  11  A     113  01  01     185  05  5      257  43 RCL     329  69 □P     401  23  23
042  52  EE    114  59  59     186  67  EQ     258  41  41     330  03  03     402  81 RST
043  04  4     115  86 STF     187  02  02     259  69 □P     331  43 RCL     403  76 LBL
044  85  +     116  05  05     188  32  32     260  04  04     332  31  31     404  15  E
045  02  2     117  61 GTO     189  06  6      261  43 RCL     333  69 □P     405  42 STO
046  95  =     118  01  01     190  67  EQ     262  40  40     334  02  02     406  00  00
047  22 INV     119  59  59     191  02  02     263  69 □P     335  04  4      407  22 INV
048  52  EE    120  29 CP      192  14  14     264  03  03     336  12  B      408  59 INT
049  69 □P     121  86 STF     193  13  C      265  07  7      337  13  C      409  52 EE
050  02  02    122  01  01     194  43 RCL     266  11  A      338  69 □P     410  04  4
051  25 CLR     123  43 RCL     195  52  52     267  12  B      339  00  00     411  22 INV
052  92 RTN     124  06  06     196  87 IFF     268  13  C      340  43 RCL     412  52 EE
053  76 LBL     125  55  ÷      197  01  01     269  43 RCL     341  30  30     413  42 STO
054  12  B     126  04  4      198  02  02     270  39  39     342  69 □P     414  06  06
055  44 SUM     127  00  0      199  02  02     271  69 □P     343  03  03     415  07  7
056  02  02    128  00  0      200  43 RCL     272  04  04     344  43 RCL     416  42 STO
057  43 RCL     129  95  =      201  51  51     273  43 RCL     345  29  29     417  04  04
058  02  02    130  22 INV     202  69 □P     274  38  38     346  69 □P     418  43 RCL
059  69 □P     131  59 INT     203  04  04     275  69 □P     347  02  02     419  00  00
060  01  01    132  67  EQ     204  43 RCL     276  03  03     348  05  5      420  59 INT
061  69 □P     133  01  01     205  50  50     277  08  8      349  12  B      421  42 STO
062  05  05    134  57  57     206  69 □P     278  11  A      350  13  C      422  05  05
063  22 INV     135  65  ×      207  03  03     279  12  B      351  43 RCL     423  44 SUM
064  97 DSZ     136  04  4      208  43 RCL     280  13  C      352  28  28     424  04  04
065  07  07    137  95  =      209  49  49     281  25 CLR     353  69 □P     425  22 INV
066  03  03    138  22 INV     210  69 □P     282  87 IFF     354  03  03     426  44 SUM
067  83  83    139  59 INT     211  02  02     283  02  02     355  43 RCL     427  00  00
068  92 RTN     140  67  EQ     212  25 CLR     284  02  02     356  27  27     428  52 EE
069  76 LBL     141  01  01     213  12  B      285  88  88     357  69 □P     429  02  2
070  13  C     142  52  52     214  13  C      286  43 RCL     358  02  02     430  22 INV
071  73 RC*     143  65  ×      215  43 RCL     287  37  37     359  06  6      431  52 EE
                                                                               432  44 SUM
                                                                               433  00  00
```

```
434  69 DP
435  20  20
436  43 RCL
437  00  00
438  36 PGM
439  20  20
440  14   D
441  42 STO
442  07  07
443  61 GTO
444  00  00
445  78  78
446  76 LBL
447  14   D
448  69 DP
449  00  00
450  43 RCL
451  20  20
452  69 DP
453  03  03
454  43 RCL
455  21  21
456  69 DP
457  02  02
458  43 RCL
459  22  22
460  69 DP
461  01  01
462  69 DP
463  05  05
464  25 CLR
465  91 R/S
466  00   0
467  00   0
468  00   0
469  00   0
470  00   0
```

Datenspeicher

```
    251331.   08        7000 30400.   34
    211714.   09        4010 00000.   35
    103546.   10        6000 30300.   36
    133335.   11        3120 00000.   37
    101324.   12        5000 30200.   38
    254131.   13        3090 00000.   39
    254127.   14        4000 30100.   40
    134122.   15        3080 00000.   41
    161733.   16        3000 21200.   42
    122637.   17        3070 00000.   43
    113242.   18        2000 20900.   44
    161746.   19        3060 00000.   45
16001170035.  20        1000 20800.   46
  170 03100.  21        3050 00000.   47
26001130027.  22        3050 00402.   48
   20 50003.  23             30000.   49
  200 30900.  24       12000 20700.   50
   20 40003.  25        3040 00401.   51
  100 30800.  26        3040 00000.   52
   20 30002.  27       36320 00000.   53
 1200 30700.  28       30320 00000.   54
   20 20002.  29       16240 00000.   55
  900 30600.  30       30240 00000.   56
   20 10002.  31       16320 00000.   57
  800 30500.  32       31350 00000.   58
  402 00000.  33       36130 00000.   59
```

Beispiel

```
      1944.      NOV
SO      5 12 19 26
MO      6 13 20 27
DI      7 14 21 28
MI   1  8 15 22 29
DO   2  9 16 23 30
FR   3 10 17 24
SA   4 11 18 25
      1944.      DEZ
SO      3 10 17 24 31
MO      4 11 18 25
DI      5 12 19 26
MI      6 13 20 27
DO      7 14 21 28
FR   1  8 15 22 29
SA   2  9 16 23 30
      1945.      JAN
SO      7 14 21 28
MO   1  8 15 22 29
DI   2  9 16 23 30
MI   3 10 17 24 31
DO   4 11 18 25
FR   5 12 19 26
SA   6 13 20 27
      1945.      FEB
SO      4 11 18 25
MO      5 12 19 26
DI      6 13 20 27
MI      7 14 21 28
DO   1  8 15 22
FR   2  9 16 23
SA   3 10 17 24
      1945.      MRZ
SO      4 11 18 25
MO      5 12 19 26
DI      6 13 20 27
MI      7 14 21 28
DO   1  8 15 22 29
FR   2  9 16 23 30
SA   3 10 17 24 31
      1945.      APR
SO   1  8 15 22 29
MO   2  9 16 23 30
DI   3 10 17 24
MI   4 11 18 25
DO   5 12 19 26
FR   6 13 20 27
SA   7 14 21 28
      1945.      MAI
SO      6 13 20 27
MO      7 14 21 28
DI   1  8 15 22 29
MI   2  9 16 23 30
DO   3 10 17 24 31
FR   4 11 18 25
SA   5 12 19 26
```

```
      1945.      JUN
SO      3 10 17 24
MO      4 11 18 25
DI      5 12 19 26
MI      6 13 20 27
DO      7 14 21 28
FR   1  8 15 22 29
SA   2  9 16 23 30
      1945.      JUL
SO   1  8 15 22 29
MO   2  9 16 23 30
DI   3 10 17 24 31
MI   4 11 18 25
DO   5 12 19 26
FR   6 13 20 27
SA   7 14 21 28
      1945.      AUG
SO      5 12 19 26
MO      6 13 20 27
DI      7 14 21 28
MI   1  8 15 22 29
DO   2  9 16 23 30
FR   3 10 17 24 31
SA   4 11 18 25
      1945.      SEP
SO      2  9 16 23 30
MO      3 10 17 24
DI      4 11 18 25
MI      5 12 19 26
DO      6 13 20 27
FR      7 14 21 28
SA   1  8 15 22 29
      1945.      OKT
SO      7 14 21 28
MO   1  8 15 22 29
DI   2  9 16 23 30
MI   3 10 17 24 31
DO   4 11 18 25
FR   5 12 19 26
SA   6 13 20 27
      1945.      NOV
SO      4 11 18 25
MO      5 12 19 26
DI      6 13 20 27
MI      7 14 21 28
DO   1  8 15 22 29
FR   2  9 16 23 30
SA   3 10 17 24
      1945.      DEZ
SO      2  9 16 23 30
MO      3 10 17 24 31
DI      4 11 18 25
MI      5 12 19 26
DO      6 13 20 27
FR      7 14 21 28
SA   1  8 15 22 29
```

Heino Velder

CBM 2001 wird vielseitiger

Seit der CBM auf den Markt gekommen ist, wurden — zumeist von Hobbyprogrammierern — eine Vielzahl von Maschinenprogrammen geschrieben, die das Arbeiten mit den Mikrocomputern wesentlich erleichtern.
Leider ist die Anwendung dieser Programme oft sehr umständlich, so daß es nahe liegt, diese Hilfsprogramme mit einem Namen zu versehen, und damit den BASIC-Befehlssatz zu erweitern.
Zuerst wird hier ein Programm vorgestellt, mit dem man den Befehlssatz des CBM sehr einfach erweitern kann.
Anschließend folgen drei Programme, die nochmals die Technik erläutern, und die zugleich eine interessante Erweiterung darstellen.
Das Maschinenprogramm liegt im Puffer des 2. Kassettenrecorders. Es kann jedoch sehr leicht in einen anderen Speicherbereich versetzt werden, da nur die drei Zeiger in der Tabelle geändert werden müssen.
Sobald man den CBM einschaltet, wird eine Routine in den Bereich der ZERO-Page geschrieben (GETCHR), mit der schrittweise das BASIC-Programm ausgelesen wird.
Diese Routine kann man verzweigen ($033A—$0344). Dabei wird — bevor das Byte an den Interpreter weitergeleitet wird — geprüft ($035A—$036A, $0376—$038), ob es sich um einen neuen Befehl handelt. Ist der Vergleich positiv verlaufen, so wird die zu dem Befehl gehörende Adresse in die Speicherzellen $01, 02 geschrieben. Mit JSR $0000 wird nun das entsprechende Maschinenprogramm ausgeführt.
In $035C ist die Anzahl der Befehle festgelegt, $00=1, $08=2, $10=3, $18=4 usw. In dem Assemblerlisting wurden vier Befehle definiert, wobei unbesetzte Stellen ($03C8—$03D8) mit $01 am Anfang gekennzeichnet werden müssen.
Die Programmierung ist recht einfach. Hierzu schreibt man den neuen Befehl (max. 5 Zeichen) in die linke obere Bildschirmecke. Dann holt man — mit einem Monitor — den entsprechenden Code, und speichert diesen (+$40) in einer Zeile der Tabelle ab. Danach muß eine Null zur Trennung folgen, und zuletzt die Startadresse (low, high-Byte) des zugehörigen Maschinenprogramms (vergl. Beispiel zu „WHITE").
Bei dem ersten Beispiel wurde der BASIC-Befehl „WHITE" definiert. Normalerweise würde die Eingabe von „WHITE" zu einem SYNTAX ERROR führen. Haben Sie das Programm zuvor mit SYS 826 eingeschaltet, so wird der Bildschirminhalt invertiert. Mit einer FOR-NEXT-Schleife kann man diesen Befehl dazu benutzen, zum z.B. das Ende einer längeren Rechenoperation anzuzeigen (siehe auch unten).
Das zweite Beispiel (Befehl „COPY") leitet alle Schriftzeichen, die an den Bildschirm gehen, sogleich an den Drucker. Somit kann man sehr schnell eine Programmbeschreibung oder Rechenergebnisse ausdrucken, ohne daß man den betreffenden Programmteil nach PRINT-Anweisungen durchforsten muß.

Einen Nachteil hat das Programm jedoch noch. Folgt ein neuer BASIC-Befehl direkt auf eine Zeilennummer, so wird die Eingabe nicht akzeptiert.

Der letzte — und sicherlich interessanteste — Zusatzbefehl (DRAW) ermöglicht es, auf sehr einfache Weise (mit Hilfe der Blockgraphik) Funktionen zu plotten.

Erweiterung des CBM-Befehlssatzes

```
033A:  A2 06     LDX  =$06     ┐
033C:  BD B7 03  LDA  $03B7,X  |SYS 826 -> schreibt neue
033F:  95 C1     STA  $C1,X    |           CHRGET-Routine
0341:  CA        DEX           |
0342:  D0 F8     BNE  $033C    |(Programm einschalten)
0344:  60        RTS           ┘
0345:  A2 06     LDX  =$06     ┐
0347:  BD B4 E0  LDA  $E0B4,X  |SYS 837 -> CHRGET-Routine
034A:  95 C1     STA  $C1,X    |           richtig stellen
034C:  CA        DEX           |
034D:  D0 F8     BNE  $0347    |(Programm ausschalten)
034F:  60        RTS           ┘
0350:  E6 C9     INC  $C9      ┐
0352:  D0 02     BNE  $0356    |Zeiger um eins erhöhen
0354:  E6 CA     INC  $CA      |
0356:  98        TYA           |und Register 'retten'
0357:  48        PHA           |
0358:  8A        TXA           |
0359:  48        PHA           ┘
035A:  A0 00     LDY  =$00     ┐
035C:  A9 18     LDA  =$18     |Prüft ob erster Buchstabe
035E:  AA        TAX           |
035F:  B1 C9     LDA  ($C9),Y  |mit neuem Befehl
0361:  DD C0 03  CMP  $03C0,X  |
0364:  F0 08     BEQ  $036E    ←übereinstimmt
0366:  8A        TXA           |
0367:  38        SEC           |
0368:  E9 08     SBC  =$08     |
036A:  B0 F2     BCS  $035E    ┘
036C:  90 1D     BCC  $038B    ← Nein, dann Routine beenden
036E:  86 00     STX  $00      ← $00 = Zeiger auf den Befehl
0370:  A5 C9     LDA  $C9      ┐
0372:  48        PHA           |Zeiger in den BASIC-Text
0373:  A5 CA     LDA  $CA      |'retten'
0375:  48        PHA           ┘
0376:  E8        INX           ┐
0377:  20 AE 03  JSR  $03AE    |Prüft ob die restlichen Buch-
037A:  BD C0 03  LDA  $03C0,X  |
037D:  C9 00     CMP  =$00     |staben mit dem neuen Befehl
037F:  F0 0F     BEQ  $0390    |
0381:  D1 C9     CMP  ($C9),Y  |übereinstimmen (max 5)
0383:  F0 F1     BEQ  $0376    ┘
```

Befehle 2. Teil

```
0385:   68          PLA             ┐
0386:   85 CA       STA   $CA       | BASIC-Zeiger zurück
0388:   68          PLA             |
0389:   85 C9       STA   $C9       | ebenfalls Register
038B:   68          PLA             |
038C:   AA          TAX             | und im BASIC-Programm
038D:   68          PLA             |
038E:   A8          TAY             | weitergehen
038F:   60          RTS             ┘
0390:   68          PLA             :
0391:   68          PLA             :
0392:   A5 00       LDA   $00       ┐
0394:   18          CLC             | X-Register auf Sprung-
0395:   69 05       ADC   =$05      | Tabelle stellen
0397:   AA          TAX             ┘
0398:   A9 4C       LDA   =$4C      ← 4C für Jump
039A:   85 00       STA   $00       :
039C:   E8          INX             :
039D:   BD C0 03    LDA   $03C0,X   ← Sprungadresse (low-Byte)
03A0:   85 01       STA   $01       :
03A2:   E8          INX             :
03A3:   BD C0 03    LDA   $03C0,X   ← Sprungadresse (high-Byte)
03A6:   85 02       STA   $02       :
03A8:   20 00 00    JSR   $0000     ┐ Dem Befehl zugehörigen
03AB:   18          CLC             | Programmteil als Subroutine
03AC:   90 DD       BCC   $038B     ┘ ausführen
03AE:   E6 C9       INC   $C9       ┐
03B0:   D0 02       BNE   $03B4     | Subroutine 'Zeiger erhöhen'
03B2:   E6 CA       INC   $CA       | (intern)
03B4:   60          RTS             ┘
03B5:   00          BRK             :
03B6:   00          BRK             :
03B7:   00          BRK             :
03B8:   20 50 03    JSR   $0350     ┐
03BB:   EA          NOP             | Daten für GETCHR-Routine
03BC:   EA          NOP             | (siehe Programmanfang)
03BD:   EA          NOP             ┘
03BE:   00          BRK             :
03BF:   00          BRK             :
```

Befehle (Anleitung)

```
 Datentabelle um neue Befehle einzufügen


.   M 03C0,03D8
          0  1  2  3  4  5  6  7
.:  03C0 57 48 49 54 45 00 E0 03
.:  03C8 01 00 00 00 00 00 00 00 Nr.0-4 für die Befehle
.:  03D0 01 00 00 00 00 00 00 00 Nr.5   dient zur Trennung
.:  03D8 01 00 00 00 00 00 00 00 Nr.6-7 Sprungadresse
                                        (low, high-Byte)


 Beispiel um den Befehl 'WHITE' zu definieren

WHITE

.   M 8000,8000
          0  1  2  3  4  5  6  7
.:  8000 17 08 09 14 05 20 20 20 ← Code auf dem Bildschirm
.   M 03C0,03C0
          0  1  2  3  4  5  6  7
.:  03C0 57 48 49 54 45 00 E0 03 ← $40 addieren und eingeben
                                   Nr. 6, 7 zeigen auf das
                                   Beispielprogramm
```

Beispiel:
10 WHITE wird nicht ins Programm aufgenommen. Man muß dann entweder das Programm mit SYS 837 ausschalten, oder den Befehl mit anderen Eingaben verbinden. So wird
10 :WHITE oder 10 FOR A=0 TO 100:NEXT:WHITE:GOTO 10 angenommen
und bewirkt, daß der Bildschirm blinkt (z.B. als Warnung).
Damit das Programm auch auf CBM-Geräten arbeitet, müssen folgende Adressen geändert werden:
PET $C9, CA/$E0B4
CBM $77, 78/$E0F8

Beispielprogramm

```
03E0:   A2 04       LDX   =$04
03E2:   A9 00       LDA   =$00
03E4:   85 40       STA   $40
03E6:   A9 80       LDA   =$80
03E8:   85 41       STA   $41
03EA:   B1 40       LDA   ($40),Y
03EC:   49 80       EOR   =$80
03EE:   91 40       STA   ($40),Y
03F0:   C8          INY
03F1:   D0 F7       BNE   $03EA
03F3:   E6 41       INC   $41
03F5:   CA          DEX
03F6:   D0 F2       BNE   $03EA
03F8:   60          RTS
```

```
03E0:   A9 03       LDA   =$03
03E2:   CD 64 02    CMP   $0264
03E5:   F0 04       BEQ   $03EB
03E7:   8D 64 02    STA   $0264
03EA:   60          RTS
03EB:   A9 04       LDA   =$04
03ED:   AA          TAX
03EE:   8D 42 02    STA   $0242
03F1:   8D 4C 02    STA   $024C
03F4:   A9 01       LDA   =$01
03F6:   8D 62 02    STA   $0262
03F9:   20 C9 FF    JSR   $FFC9
03FC:   60          RTS
```

```
COPY    (DEFINIEREN)

.   M 8000,8000
          0  1  2  3  4  5  6  7
.:  8000 03 0F 10 19 20 20 20 20
.   M 03C0,03C0
          0  1  2  3  4  5  6  7
.:  03C0 43 4F 50 59 00 00 E0 03
```

Frank Wähner

Blick in die CPU-Register des CBM 2001

Beschreibung

Mit diesem Programm ist es möglich, (nach der Initialisierung durch SYS8026), Maschinenprogramme durch Drücken der Shift-Return-Taste zu unterbrechen. Gleichzeitig wird der Inhalt sämtlicher CPU-Register auf dem Bildschirm ausgegeben. Es kann zwischen hexa- und dezimaler Zahlenausgabe gewählt werden. Dezimal, wenn in der Speicherstelle 8191 eine 0 steht, sonst hexadezimal. Unterbrechung erfolgt auch durch den Maschinenbefehl „BRK".

Programmbeschreibung

Funktionsprinzip: Falls das Interrupt-Disableflag nicht gesetzt ist (nur dann ist das Programm in der Lage, eine Unterbrechung durchzuführen), löst ein Timer in der Sekunde 60mal einen Interrupt aus, der folgendermaßen aussieht:

Nach der „Rettung" der CPU-Register auf dem Stack durchläuft der Computer anschließend eine Tastaturabfrageroutine. Startadresse: 59013 (steht in 537,538).

Genau hier setzt das Unterbrechungsprogramm an. Der Computer wird einfach umgeleitet, d.h., er muß einen Umweg über das Unterbrechungsprogramm machen (das ist auch der Grund, warum die Kassettenladeroutine nicht mehr funktioniert (Zeitverzögerung)). Das Unterbrechungsprogramm überprüft nun, ob die Shift-Return-Taste gedrückt ist. Ist dies nicht der Fall, wird das Interruptprogramm normal fortgesetzt (Sprung nach 59013). Andernfalls erfolgt ein Ausdruck sämtlicher CPU-Register, eine Korrektur des Stackregisters und ein anschließender Rücksprung in den Direktmodus.

Anwendung des Programms

Laden des Programms durch Shift-Run, NEW, Laden des zu untersuchenden Maschinenprogramms, Initialisierung des Unterbrechungsprogramms durch SYS8026. Abschalten durch SYS8040 (unbedingt vor Kassettenoperationen).

Unterprogramm Zahlenumwandlung bzw. Ausgabe

Die Binärzahl steht in 8189/8190.

Wird dezimale Ausgabe gewünscht, so steht in 8191 eine 0.

Aufruf durch SYS7921.

Ausgabe: Zeichenkette (Zahl) + ein „Space" (Unterdrückung führender Nullen).

```
IRQ+BRK

READY.

 4 DATA173,255,31,240,3,76,21,31,76,255,30,32,230,30,160,1,152,32,39
 8 DATA202,169,32,76,234,227,173,254,31,174,253,31,133,177,134,178,162,144,56
 12 DATA32,27,219,32,175,220,169,1,96,169,36,160,0,140,3,1,140,5,1
 16 DATA200,141,0,1,173,254,31,240,3,32,56,31,173,253,31,32,56,31,169
 20 DATA0,200,153,0,1,96,141,251,31,152,74,173,251,31,144,4,74,74,74
 24 DATA74,41,15,24,105,48,201,58,144,2,105,6,153,0,1,200,152,74,144
 28 DATA228,96,120,160,3,185,214,31,153,25,2,136,16,247,88,96,120,169,133
 32 DATA141,25,2,169,230,141,26,2,88,96,173,4,2,240,89,173,252,31,141
 36 DATA251,31,174,3,2,142,252,31,224,27,208,73,201,255,208,69,169,0,141
 40 DATA254,31,186,138,24,105,6,72,162,4,138,72,10,10,105,229,160,31,32
 44 DATA39,202,104,170,104,141,253,31,138,72,32,241,30,104,170,202,16,229,169
 48 DATA218,160,31,32,39,202,104,141,253,31,104,141,254,31,32,241,30,169,13
 52 DATA32,234,227,32,132,197,76,141,195,76,133,230,117,31,142,31,13,66,82
 56 DATA69,65,75,32,73,78,32,0,13,80,58,0,13,65,58,0,13,88,58
 60 DATA0,13,89,58,0,13,83,58,0,99,126,255,255
 1000 POKE134,230:POKE135,30
 1010 FORI=7910TO8188:READP:POKEI,P:NEXT:END
 1995 AD=7909:I=0
 19910 I=I+4:Q=0:PRINT:PRINTI;"DATA";
 19915 AD=AD+1:IFAD>8188THENSTOP
 19920 D$=STR$(PEEK(AD))
 19930 D$=RIGHT$(D$,LEN(D$)-1)
 19935 IFQ<18THEND$=D$+","
 19942 PRINTD$;:Q=Q+1:REM
 19949 IFQ>18GOTO19910
 19950 GOTO19915
 READY.
```

Oliver Fischer

CBM-Maschinencode im String gespeichert

In verschiedenen BASIC-Dialekten, z.B. in dem des TRS-80 oder des DAI gibt es den recht leistungsfähigen BASIC-Befehl „VARPTR". Dieser Befehl dient dazu, die Anfangsadresse einer beliebigen BASIC-Variablen im Speicher festzustellen. Leider fehlt dieser Befehl in den BASIC-Dialekten des CBM oder PET. Folgendes Maschinenprogramm dient nun dazu, diesen Befehl auf CBM-Computern nachzubilden. Besitzer des PET müßten, um das Programm auf ihren Computern zum Laufen zu bringen, einige Unterprogrammsprungadressen ändern. Eine Auflistung der Unterprogramme ist im Heft „Microcomputer-Anwendungen", erschienen als Sonderheft im Franzis-Verlag, abgedruckt worden.

Nun zu einer Anwendung dieses Befehls. Schreibt man ein kleines Maschinenprogramm, so legt man es normalerweise in den Cassettenpuffer 2. Es gibt aber auch die Möglichkeit, den BASIC-Bereich nach oben zu beschränken und somit Platz für Maschinenprogramme zu erhalten. Als weitere Alternative dazu möchte ich ein Programm vorstellen, mit dem es möglich ist, verschiebbare Maschinenprogramme mit bis zu 255 Bytes Länge in BASIC-Stringvariablen abzuspeichern und später mit SYS wieder starten zu können. Dazu wird zunächst eine Stringvariable mit soviel Zeichen definiert wie das Maschinenprogramm Bytes hat. Anschließend wird mit dem simulierten VARPTR-Befehl und dem Unterprogramm ab Zeile 100 der Beginn der Stringvariable festgestellt und mittels einer FOR-NEXT-Schleife die Bytes des Maschinenprogramms in die Variable „gepoked". Es kann dann ein beliebiges BASIC-Programm folgen.

Will man nun das Maschinenprogramm wieder starten, stellt man mit dem simulierten VARPTR-Befehl und Unterprogramm 100 wieder die Startadresse der Stringvariable fest und startet das Maschinenprogramm mit SYS (Anfangsadresse der Stringvar.). Als Beispiel wurde ein Maschinenprogramm verwendet, das bei jedem Aufruf den Bildschirm auf reverse und umgekehrt umschaltet.

Achtung

Man darf bei der Einrichtung der Stringvariablen nicht vergessen, " " zu addieren, da sonst die Stringvariable nicht in den Stringbereich kopiert wird, sondern im BASIC-Text stehenbleibt, was dann bei verschiedenen Maschinencodes zu Fehlern führt, z.B. wird Code 0 (BRK) als Zeilenende interpretiert.

Maschinenprogramm für die Simulierung des VARPTR-Befehls

```
033A A0 00     LDY =$00
033C B1 77     LDA($77),Y / C9
033E C9 2C     CMP =$2C
0340 D0 35     BNE $0377
0342 20 78 03  JSR $0378
0345 20 6D CF  JSR $CF6D / CF7B
0348 A2 02     LDX =$02
034A B5 43     LDA $43,X/95
034C 9D 7E 03  STA $037E,X
034F CA        DEX
0350 D0 F8     BNE $034A
0352 A0 00     LDY =$00
0354 B1 77     LDA($77),Y/C9
0356 C9 2C     CMP =$2C
0358 D0 1D     BNE $0377
035A 20 78 03  JSR $0378
035D 20 6D CF  JSR $CF6D/CF7B
0360 AD 80 03  LDA $0380
0363 85 5F     STA $5F/B1
0365 AD 7F 03  LDA $037F
0368 85 60     STA $60/B2
036A A2 90     LDX =$90
036C 38        SEC
036D 20 55 DB  JSR $DB55/DB1B
0370 A6 44     LDX $44/96
0372 A4 45     LDY $45/97
0374 20 E3 DA  JSR $DAE3/DAA6
0377 60        RTS
0378 E6 77     INC $77/C9
037A D0 02     BNE $037E
037C E6 78     INC $78/CA
037E 60        RTS
```

Beispiel für die Anwendung des simulierten VARPTR-Befehls

```
 10 REM   FUER NUMERISCHE VA-
          RIABLE   NUR   SYS
          (826), GESUCHTE VA-
          RIABLE,  SPEICHER-
          VARIABLE; B ENTHA-
          ELT ADRESSE DER VA-
          RIABLE
 20 REM   FUER   STRINGVARI-
          ABLE UNTERPROGRAMM
          100 VERWENDEN AD-
          =ADRESSE; L=LAENGE
          DER STRINGVARIABLE
 30 A=4
 40 SYS(826),A,B
 50 PRINT"VARIABLE A STEHT AB
    ADRESSE";B
 60 GOSUB 100
 70 PRINT "STRINGVARIABLE A$
          STEHT          AB
          ADRESSE";AD;"UND
          HAT DIE LAENGE";L
 80 END
100 SYS(826),A$,B
110 L=PEEK(B)
120 LB=PEEK(B+1):HB=PEEK
    (B+2)
130 AD=LB+HB*256
140 RETURN
```

Programm für Speicherung von Maschinenprogrammen in Stringvariablen

Zuerst VARPTR-Maschinenprogramm laden, dann untenstehendes BASIC-Programm!

```
 10 A$="123456789012345678
    90123456"+""
 20 GOSUB 100
 30 FOR I=AD TO AD+L-1
 40 READ MC
 50 POKE I,MC
 60 NEXT
 70 GOSUB 100
 80 SYS(AD):REM      AD=START-
    ADRESSE   DES   MASCHINEN-
    PROGRAMMS
 90 END
100 SYS(826),A$,B
110 L=PEEK(B):LB=PEEK(B+1)
    :HB=PEEK(B+2)
120 AD=LB+256*HB
130 RETURN
140 DATA
    160,0,132,1,169,128,133,
    2,177,1
150 DATA
    73,128,145,1,200,208,247,
    230,2
160 DATA
    169,132,197,2,208,239,96
```

Ralf Nolting

Hexadezimal-Tastatur mit TI-58

Beschreibung

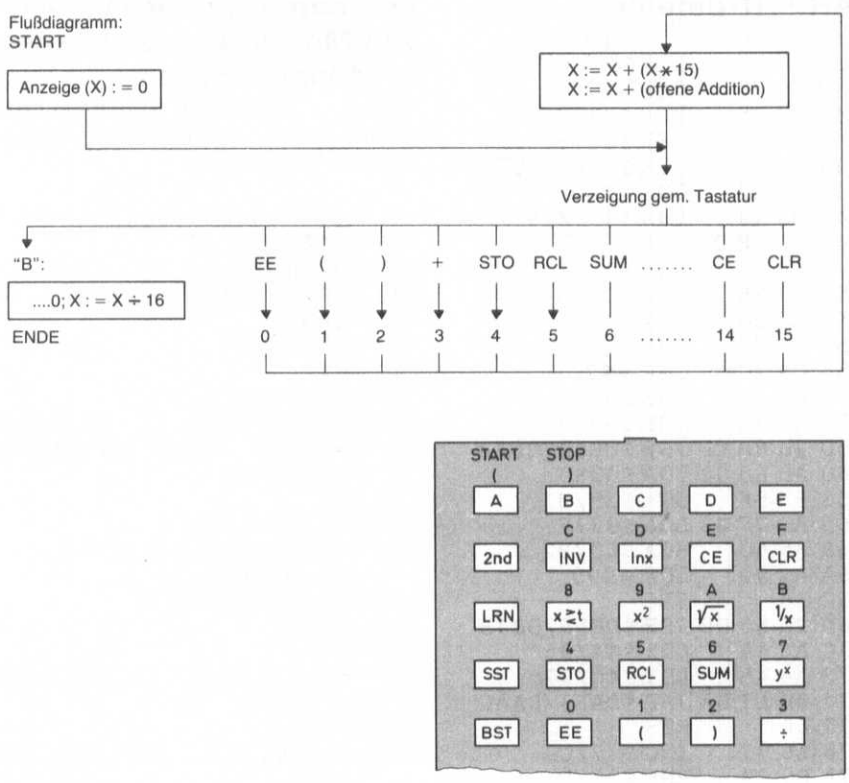
Idee:
Eingabe einer Hexadezimal-Zahl wie auf einer 16er Tastatur (eine Taste je Ziffer)
Verarbeitung: Umrechnung ins Dezimalsystem, dabei werden keine offenen Operationen berührt.

1. Verteilung für TI-59: 239.89, für TI-58: 239.29
2. Eröffnung der Eingabe mit A (entspricht „Klammer auf")
3. Eingabe der Ziffern gemäß Schablone, dabei immer blinkende Anzeige abwarten (etwa 0,5 s)
4. Abschluß mit B (entspricht „Klammer zu")

Beispiel:
Zu berechnen ist $ABC99_{16} - BF74_{16}$
Tastenfolge: (schwarz ≙ normal, grau ≙ Schablone)
A A B C 9 9 B − A B F 7 4 B =
Ergebnis: $654\,629_{10}$

Registerbelegung:
R00: Sprungadresse
3 Klammerebenen
Unterprogrammrücksprungregister,
Flags werden gelöscht

Methode:
Es handelt sich um eine einfache Eingabeschleife, bei der eine besondere Eigenschaft des TI-58/59 ausgenutzt wird: Bei SBR-Befehl am Ende des Programmspeichers bewirkt (neben einem „Error") einen Sprung zu dem Label (auch allgemeines Lbl), dessen Taste als nächstes gedrückt wird.




```
000  83 GO*     062  76 LBL
001  00  00     063  24 CE
002  76 LBL     064  01  1
003  52 EE      065  04  4
004  00  0      066  81 RST
005  81 RST     067  76 LBL
006  76 LBL     068  25 CLR
007  53  (      069  01  1
008  01  1      070  05  5
009  81 RST     071  81 RST
010  76 LBL     072  76 LBL
011  54  )      073  12  B
012  02  2      074  07  7
013  81 RST     075  09  9
014  76 LBL     076  42 STO
015  55  ÷      077  00  00
016  03  3      078  81 RST
017  81 RST     079  00  0
018  76 LBL     080  54  )
019  42 STO     081  55  ÷
020  04  4      082  01  1
021  81 RST     083  06  6
022  76 LBL     084  54  )
023  43 RCL     085  24 CE
024  05  5      086  91 R/S
025  81 RST     087  76 LBL
026  76 LBL     088  11  A
027  44 SUM     089  53  (
028  06  6      090  53  (
029  81 RST     091  02  2
030  76 LBL     092  03  3
031  45 Yˣ      093  01  1
032  07  7      094  42 STO
033  81 RST     095  00  00
034  76 LBL     096  00  0
035  32 X↕T     097  61 GTO
036  08  8      098  02  02
037  81 RST     099  39  39
038  76 LBL     100  00  0
039  33 X²      101  00  0
040  09  9
041  81 RST
042  76 LBL     220  00  0
043  34 ΓX      221  00  0
044  01  1      222  00  0
045  00  0      223  00  0
046  81 RST     224  00  0
047  76 LBL     225  00  0
048  35 1/X     226  00  0
049  01  1      227  00  0
050  01  1      228  00  0
051  81 RST     229  00  0
052  76 LBL     230  00  0
053  22 INV     231  85  +
054  01  1      232  53  (
055  02  2      233  24 CE
056  81 RST     234  65  ×
057  76 LBL     235  01  1
058  23 LNX     236  05  5
059  01  1      237  54  )
060  03  3      238  85  +
061  81 RST     239  71 SBR
```

Gerd S. Gottschling

Programmverschiebung mit dem TI-59

Beschreibung

Der programmierbare Taschenrechner TI-59 ist durch die Modul- und Magnetkartentechnik ein leistungsfähiger Rechner, mit dem nicht nur privat gearbeitet wird.
Eine Kapazitätserweiterung wäre ohne Zweifel ein Kassetteninterface oder ein frei programmierbares Leermodul.
Ein weiteres Handicap ist die „statische Programmierung", d.h., daß es bisher nicht möglich ist, Programmteile oder ganze Programme beliebig zu verschieben, ohne die mühsame Ins-Technik anwenden zu müssen.
Die im folgenden geschilderten Zusammenhänge und die daraus entwickelte Routine hebt unter Beachtung der für einige Einschränkungen geltenden Regeln dieses Manko auf.
Bekanntlich kann die Speicherkapazität des TI-59 auf Programmspeicherplätze (kurz PSP) und Datenspeicherplätze (kurz DSP) mittels der Operation 17 (n OP 17) verteilt werden. Für den Grobbereich gelten die Grenzen:
160 PSP/100 DSP und
960 PSP/ 0 DSP.
Die PSP 000 bis 159 können nicht weiter gesplittet werden, es bleiben 800 PSP für 100 DSP, d.h., jeder DSP nimmt eine Zelle mit 8 PSP auf (siehe Tabelle 1). Der nicht splittungsfähige Bereich entzieht sich einer möglichen Verschiebung, weil diese PSP kein entsprechendes DSP-Äquivalent besitzen. Die 160 PSP bleiben jedoch der Programmierung offen und werden für kleine Hilfsprogramme (kurz HPgm) genutzt, um den manuellen Arbeitsaufwand gering zu halten.
Das HPgm 1 wird für das Auffinden der DSP-Zelle mit PSP-Anfang und PSP-Ende benötigt und erspart das Nachschlagen im Handbuch. Man kann jedoch auch mit Tabelle 1 (DSP/PSP-Splitting) arbeiten.
Zur Verdeutlichung der weiteren Erklärungen wird empfohlen, zunächst die HPgme (ab PSP 000) und das Modellprogramm „Null-Druck" (ab PSP 240) bei der normalen Verteilung einzugeben.
Durch CLR und D erfolgt der Druck einer großen Null, wenn das Programm richtig übernommen worden ist (siehe Bild 5).

Vorgang

Eingabe PSP Taste A Anzeige $\lfloor DSP \rfloor$ 0 $\lfloor PSP_{Anf} \rfloor \cdot \lfloor PSP_{End} \rfloor$

Eingabe DSP Taste B Anzeige dto.

(160 ≤ PSP ≤ 952 ; 0 ≤ DSP ≤ 99)

000	76	LBL	027	85	+
001	11	A	028	09	9
002	55	÷	029	05	5
003	08	8	030	02	2
004	94	+/-	031	95	=
005	85	+	032	85	+
006	01	1	033	53	(
007	01	1	034	24	CE
008	09	9	035	85	+
009	76	LBL	036	07	7
010	12	B	037	54	)
011	75	-	038	65	×
012	59	INT	039	01	1
013	82	HIR	040	52	EE
014	04	04	041	03	3
015	95	=	042	94	+/-
016	67	EQ	043	85	+
017	00	00	044	82	HIR
018	22	22	045	14	14
019	01	1	046	65	×
020	82	HIR	047	01	1
021	34	34	048	52	EE
022	82	HIR	049	04	4
023	14	14	050	95	=
024	65	×	051	22	INV
025	08	8	052	52	EE
026	94	+/-	053	91	R/S

Bild 1
DSP/PSP-Test (HPgm 1)

Beispiele :

a) Welche DSP-Zelle enthält den PSP 243 ?
b) Welche PSP umfaßt die Zelle DSP39 ?

a) 243 A → 89 0 240.247
b) 39 B → 39 0 640.647

Als Vorbereitung für die Verschiebung wird nun die maximale DSP-Verteilung (10 OP 17) getastet. Dieser Vorgang veranlaßt den Rechner, die PSP 160 bis 959 als Zifferkombination in entspre-

(bitte umblättern)

Programmverschiebung mit dem TI-59

```
Vorgang
        Eingabe  10     Taste  OP  17     Anzeige  159.99
        Eingabe DSP     Taste  SBR SBR    Anzeige  DSP
        Eingabe Kode    Taste      R/S    Anzeige  1 00  Druck ..... EE1  DSP
        Eingabe EE2     Taste      R/S    Anzeige  0     Druck .....      DSP

        für EE1 positiv  :  EE2 = -( EE1 - 5 )
        für EE1 negativ  :  EE2 =  ( |EE1| + 5 )
```

054	76	LBL	073	95	=
055	71	SBR	074	22	INV
056	42	STO	075	59	INT
057	00	00	076	22	INV
058	91	R/S	077	52	EE
059	69	OP	078	58	FIX
060	04	04	079	07	07
061	73	RC*	080	69	OP
062	00	00	081	06	06
063	65	×	082	61	GTO
064	01	1	083	01	01
065	52	EE	084	56	56
066	95	=	085	00	0
067	69	OP	086	00	0
068	06	06			
069	65	×			
070	01	1			
071	52	EE			
072	91	R/S			

Bild 2
DSP-Druck (HPgm 2)

```
Beispiele :

a) DSP-Zelle mit positivem Exponent  EE1
b) DSP-Zelle mit negativem Exponent  EE1

a)   69   SBR SBR  →  69
     712       R/S  →  1 00    Druck :  - 6.8566814  44    69
                                        EE2 = -( 44 - 5 )
                                        EE2 =  - 39

     39   +/-  R/S  →  0       Druck :  - 0.1361250        69

b)   89   SBR SBR  →  89
     912       R/S  →  .1 00   Druck :  - 5.3852454 - 85   89
                                        EE2 = ( |- 85| + 5 )
                                        EE2 =    90

     90        R/S  →  0       Druck :  - 0.5390766        89

für EE2 > 99 : manuelle Multiplikation des 1. Wertes
               mit EE2 = - EE1 , dann Eingabe
               EE2 = 5  R/S
               (siehe DSP 84 u. DSP 75 in Tabelle 2 )
```

chende DSP zu schieben. Hierbei treten einige Besonderheiten auf.
Als Beispiel betrachte man DSP 89 mit PSP 240 bis 247 (aus HPgm 1/Beispiel a), die Darstellung der Zelle übernimmt das HPgm 2 und man erhält:
−5.3852454−85 89
−0.5390766 .
Im unteren Zeilendruck ist die 1. Nachkommastelle (5) identisch mit der 6. Nachkommastelle der oberen Zeile. Diese Darstellung wurde zur besseren Identifikation gewählt, da die 7. Nachkommastelle (4) gerundet sein kann. Die zunächst wirr erscheinende Zifferkombination ist jedoch bei näherer Betrachtung durchaus logisch aufgebaut. Im Vergleich mit dem Modellprogramm fällt auf, daß 7 der 8 PSP (PSP 241 bis PSP 247) direkt sichtbar sind und nur die Identifikation des PSP 240 schwierig ist.
Der Rechner kann leider nur 15 Informationen in Zifferform anzeigen (13 des DSP plus 2 des Exponenten). Die fehlende 16. Information muß also die Vorzeichenkombination ergeben. Die Matrix zeigt:

Zeile	DSP-Wert	Exp	16. Info
a	+	+	0/1
b	−	+	2/3
c	+	−	4/5
d	−	−	6/7
e	?	?	8/9

Info-St.		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
DSP	−	5	3	8	5	2	4	5	4					−	8	5	
						−	0	5	3	9	0	7	6	6			
Kode		5	3	8	5	2	4	5	3	9	0	7	6	6	8	5	?
PSP		247		246		245		244		243		242		241		240	
Taste		(		+		CE		(		List		Lbl		NOP		?	

Für das Beispiel trifft Zeile d (beide Vorzeichen negativ) zu, die 16. Information ist eine 6, und der PSP 240 ist also mit Kode 56, Taste Del belegt.
Die Matrix macht zwei Einschränkungen sichtbar.

1. Die 16. Information kann nicht eindeutig definiert werden. Wenn sich diese Information durch die Pro-

Tabelle 1: DSP/PSP-Splitting

	DSP	PSP_A	PSP_E
10 OP 17			159
	99	160	167
	98	168	175
	97	176	183
	96	184	191
	95	192	199
	94	200	207
	93	208	215
	92	216	223
	91	224	231
	90	232	239
	--	---	---
9 OP 17			239
	89	240	247
	88	248	255
	87	256	263
	86	264	271
	85	272	279
	84	280	287
	83	288	295
	82	296	303
	81	304	311
	80	312	319
	--	---	---
8 OP 17			319
	79	320	327
	78	328	335
	77	336	343
	76	344	351
	75	352	359
	74	360	367
	73	368	375
	72	376	383
	71	384	391
	70	392	399
	--	---	---
7 OP 17			399
	69	400	407
	68	408	415
	67	416	423
	66	424	431
	65	432	439
	64	440	447
	63	448	455
	62	456	463
	61	464	471
	60	472	479
	--	---	---
6 OP 17			479
	59	480	487
	58	488	495
	57	496	503
	56	504	511
	55	512	519
	54	520	527
	53	528	535
	52	536	543
	51	544	551
	50	552	559
	--	---	---
5 OP 17			559

	DSP	PSP_A	PSP_E
5 OP 17			559
	49	560	567
	48	568	575
	47	576	583
	46	584	591
	45	592	599
	44	600	607
	43	608	615
	42	616	623
	41	624	631
	40	632	639
	--	---	---
4 OP 17			639
	39	640	647
	38	648	655
	37	656	663
	36	664	671
	35	672	679
	34	680	687
	33	688	695
	32	696	703
	31	704	711
	30	712	719
	--	---	---
3 OP 17			719
	29	720	727
	28	728	735
	27	736	743
	26	744	751
	25	752	759
	24	760	767
	23	768	775
	22	776	783
	21	784	791
	20	792	799
	--	---	---
2 OP 17			799
	19	800	807
	18	808	815
	17	816	823
	16	824	831
	15	832	839
	14	840	847
	13	848	855
	12	856	863
	11	864	871
	10	872	879
	--	---	---
1 OP 17			879
	9	880	887
	8	888	895
	7	896	903
	6	904	911
	5	912	919
	4	920	927
	3	928	935
	2	936	943
	1	944	951
	0	952	959
	--	---	---
0 OP 17			959

grammierung zu einer ungeraden Ziffer ergibt, so wandelt der Rechner bei der Programmverschiebung die ungerade Ziffer in die nächst kleinere gerade Ziffer um.
Bei der Programmierung dürfen die ersten PSP einer jeden Zelle nur geradzahlige Kodes beinhalten.

2. Für eine 8 oder 9 als 16. Information findet sich keine Vorzeichenzugehörigkeit. Wird eine solche DSP-Zelle aufgerufen, so blinkt die Anzeige, und die PSP dieser Zelle sind nach der Programmverschiebung verfälscht.
Bei der Programmierung dürfen die Kodes der PSP einer jeden Zelle nicht mit 8 oder 9 enden.

Eine weitere Einschränkung für das Gelingen der Verschiebung ist der Verzicht auf die direkte Adressierung wie GTO nnn oder SBR nnn.
Bei der Programmierung bzw. der Programmerstellung kann man darauf achten, daß die kritischen PSP mit einem funktionslosen Kode belegt sind. Dieser muß die o.a. Bedingungen erfüllen. Hierfür eignen sich 26 (2nd), 46 (Ins) und 56 (Del), sie lassen sich allerdings nur mit der bekannten STO-BST-Del-Methode einfügen.
Weitere kritische Stellen sind die letzten PSP einer jeden Zelle, wenn dort Kodes mit führenden Nullen (00 bis 09) programmiert sind. Durch den DSP-Aufruf wird der Wert des Exponenten und somit die 14. und 15. Information verändert. In solchen Fällen müssen ebenfalls die o.a. Befehle oder der Befehl 68 (NOP) programmiert werden.
Die eigentliche Programmverschiebung ist bei Beachtung der Besonderheiten leicht zu realisieren.
Wie bereits erwähnt, wird durch 10 OP 17 der Rechner zur DSP-Zellenkodierung veranlaßt. Mit dem HPgm 1 ist festgestellt worden, daß das Modellprogramm die DSP 89 bis 69 belegt.
Um das Programm nun z.B. nach PSP 480 zu verschieben, werden mit dem HPgm 1 die dann belegten DSP bestimmt, es sind dies die DSP 59 bis 39. Nun müssen die DSP ausgetauscht werden, diese Arbeit übernimmt das dafür erstellte HPgm 3. Wird der Gesamtaustausch in diesem Programm benutzt, so sind durch die fallende Austauschfolge (DSP 89 nach DSP 59, DSP

(bitte umblättern)

Programmverschiebung mit dem TI-59

88 nach DSP 58 usw.) richtungsgleiche Überschneidungen ausgeschlossen, der Austausch DSP 89 nach DSP 90, DSP 88 nach DSP 89 usw. ist jedoch erlaubt.
Ist der DSP-Austausch abgeschlossen, so kann zur Prüfung HPgm 2 verwendet werden.
Abschließend ändert man nur noch die Verteilung mit n OP 17, so daß der Rechner die DSP-Inhalte in PSP dekodiert. Die Eingabe für n ist so zu wählen, daß der dann höchste belegte PSP 647 innerhalb der Endverteilung liegt.
Für das Beispiel kann n gleich 2 oder 3 getastet werden, n gleich 0 oder 1 ist

Vorgang
Eingabe 10 Taste OP 17 Anzeige 159.99

für Gesamt-DSP-Austausch
Eingabe DSP_v . DSP_n Taste E Anzeige 0.

für Einzel-DSP-Austausch
Eingabe DSP_1 . DSP_2 Taste SBR GTO Anzeige 0.

Index v : vor der Verschiebung max. belegte Zelle
n : nach dem Austausch max. belegte Zelle
1 : zu übertragende Zelle
2 : aufnehmende Zelle

Beispiel : DSP-Austausch des Modell-Programms „Null — Druck"
(DSP_v = 89 , DSP_n = 59)

89 . 59 E ⟶ 0

(Prüfung kann mit „DSP-Druck" erfolgen)

Tabelle 2: DSP-Druck

vor der Verschiebung		nach der Verschiebung	
-5.3852454-85	89	-5.3852454-85	59
-0.5390766	89	-0.5390766	59
1.0695406-52	88	1.0695406-52	58
0.0593067	88	0.0593067	58
-7.6920043-55	87	-7.6920043-55	57
-0.4254028	87	-0.4254028	57
-9.2026995 54	86	-9.2026995 54	56
-0.9500436	86	-0.9500436	56
-6.86895-47	85	-6.86895-47	55
-0.0043654	85	-0.0043654	55
-9.2006906-95	84	-9.2006906-95	54
-0.0569036	84	-0.0569036	54
-1.1420002-37	83	-1.1420002-37	53
-0.0190711	83	-0.0190711	53
-2.3119739-15	82	-2.3119739-15	52
-0.3876227	82	-0.3876227	52
-7.6689298-15	81	-7.6689298-15	51
-0.9825227	81	-0.9825227	51
-9.2247143 32	80	-9.2247143 32	50
-0.4271024	80	-0.4271024	50
-7.6683861-37	79	-7.6683861-37	49
-0.6130712	79	-0.6130712	49
7.6689245-32	78	7.6689245-32	48
0.4471044	78	0.4471044	48
7.6689235 13	77	7.6689235 13	47
0.3471297	77	0.3471297	47
5.6689245-33	76	5.6689245-33	46
0.4471014	76	0.4471014	46
-9.2684271-97	75	-9.2684271-97	45
-0.7103432	75	-0.7103432	45
-5.6686801-47	74	-5.6686801-47	44
-0.0142011	74	-0.0142011	44
-6.8680242-15	73	-6.8680242-15	43
-0.4201000	73	-0.4201000	43
-5.668-15	72	-5.668-15	42
0.0000000	72	0.0000000	42
-4.6010001 34	71	-4.6010001 34	41
-0.0100010	71	-0.0100010	41
-5.6000068-85	70	-5.6000068-85	40
-0.6801006	70	-0.6801006	40
-6.8566814 44	69	-6.8566814 44	39
-0.1361250	69	-0.1361250	39

nicht möglich, weil das Modellprogramm u.a. DSP 11 benutzt.
Die Anweisung GTO 480 LRN oder List zeigt, daß die Programmverschiebung durchgeführt worden ist. Als Endtest wird der Nulldruck mit CLR,D gefordert.
Um den Vorgang noch einmal in Kurzform darzulegen, soll das Programm nun von PSP 480 nach PSP 160 verschoben werden.

Zusammenfassung

Die beschriebene Routine ermöglicht unter Beachtung der „Verschieberegeln" eine bisher nicht mögliche Programmverschiebung. Die dazu notwendige sorgfältige Programmierung ist nicht besonders aufwendig und benötigt nur wenige zusätzliche PSP. Mit verschobenen Programmen kann normal gearbeitet werden, so sind nachträgliche Änderungen, Schreiben auf Magnetkarte usw. möglich.
Die gewonnenen Kenntnisse tragen meiner Meinung nach dazu bei, den Einsatzbereich des Rechners zu erweitern. Es ist beispielsweise eine „Magnetkartenbank!" realisierbar, die bei Bedarf Programme bzw. Programmteile „verleiht" und „mietet" oder den Rechner zu programmieren, ohne den TI-59 in den LRN-Modus zu schalten. Diese Technik der Programmeingabe sei abschließend an einem kleinen Beispiel erklärt.

085	76	LBL	106	42	STO	120	76	LBL
086	15	E	107	00	00	121	61	GTO
087	71	SBR	108	22	INV	122	71	SBR
088	01	01	109	59	INT	123	01	01
089	06	06	110	65	×	124	06	06
090	67	EQ	111	01	1	125	63	EX*
091	01	01	112	00	0	126	01	01
092	56	56	113	00	0	127	63	EX*
093	71	SBR	114	95	=	128	00	00
094	01	01	115	42	STO	129	25	CLR
095	25	25	116	01	01	130	92	RTN
096	69	OP	117	73	RC*	131	00	0
097	30	30	118	00	00			
098	69	OP	119	92	RTN			
099	31	31						
100	71	SBR						
101	01	01						
102	17	17						
103	61	GTO						
104	00	00						
105	90	90						

Bild 3: DSP-Austausch (HPgm 3)

Instruktionen

1. Verteilung ändern
   CMs 10 OP 17 _____ 159.99
2. DSP_v und DSP_n bestimmen
   a) PSP 480 A _____ 590480.487 (DSP_v)
   b) PSP 160 A _____ 990160.167 (DSP_n)
3. Austausch der DSP
   59.90 E _____ 0.
4. Austausch-Test (nicht zwingend)
   ……z.B. für DSP 99 = DSP 59 (HPgm 2)
5. Endverteilung
   Achtung!
   Der letzte PSP der Endverteilung muß größer oder gleich dem letzten PSP des verschobenen Programms sein.

   hier: PSP 327 nach der Verschiebung
   z.B. 7 OP 17 _____ 399.69
6. Null-Druck-Test
   CLR D _____ 0.Druck

vorgang

Eingabe	10	Taste	OP 17	Anzeige	159.99		
Eingabe	DSP	Taste	SBR RST	Anzeige	DSP		
Eingabe	I (VZ)	Taste	R/S	Anzeige	0.00		
Eingabe	II (VZ)	Taste	R/S	Anzeige	1 00		
Eingabe	III (VZ EE)	Taste	R/S	Anzeige	0		

für I : a.bcdefghij (ersten 10 Info mit Vorzeichen nach Matrix)
für II : k.lm (Info 11;12;13 mit ...)
für III : EE (Info 14;15 mit ... für Exp. nach ...)

Beispiel: Geben Sie bitte folgendes HPgm 4 ein:

	10	OP	17	159.99
		CMs	CLR	0
DSP	99	SBR	RST	99.
I	9.76800840 +/-		R/S	0.00
	0			
II	7.31 +/-		R/S	1 00
III	67 +/-		R/S	0
DSP	98	SBR	RST	98.
I	9.10048040		R/S	0.00
	5			
II	6.91		R/S	1 00
III	60		R/S	0
	3017313773	STO	04	
	3327131517	STO	03	
	4500162436	STO	02	
	7323133333	STO	01	
	4	STO	00	
		OP	17	639.39
		CLR	A'	Druck

131	76	LBL	145	01	1	159	91	R/S
132	81	RST	146	00	0			
133	42	STO	147	94	+/-			
134	00	00	148	95	=			
135	91	R/S	149	74	SM*			
136	72	ST*	150	00	00			
137	00	00	151	01	1			
138	25	CLR	152	52	EE			
139	58	FIX	153	91	R/S			
140	02	02	154	64	PD*			
141	91	R/S	155	00	00			
142	65	×	156	22	INV			
143	01	1	157	58	FIX			
144	52	EE	158	25	CLR			

Bild 4: DSP-Speicher (HPgm 4)

(bitte umblättern)

Programmverschiebung mit dem TI-59

∘	240	56	DEL	300	38	SIN	∘360	76	LBL
	241	68	NOP	301	97	DSZ	361	14	D
	242	76	LBL	302	11	11	362	01	1
	243	90	LST	•303	23	LNX	363	42	STO
	244	53	(	∘304	56	DEL	364	01	01
	245	24	CE	305	71	SBR	365	68	NOP
	246	85	+	306	22	INV	366	68	NOP
•	247	53	(	307	25	CLR	•367	56	DEL
∘	248	24	CE	308	98	ADV	∘368	56	DEL
	249	75	-	309	92	RTN	369	01	1
	250	06	6	310	68	NOP	370	00	0
	251	93	.	•311	76	LBL	371	01	1
	252	05	5	∘312	22	INV	372	42	STO
	253	54	)	313	43	RCL	373	02	02
	254	69	OP	314	02	02	374	68	NOP
•	255	10	10	315	71	SBR	•375	68	NOP
∘	256	56	DEL	316	42	STO	∘376	56	DEL
	257	85	+	317	71	SBR	377	01	1
	258	02	2	318	24	CE	378	00	0
	259	54	)	•319	92	RTN	379	00	0
	260	42	STO	∘320	76	LBL	380	00	0
	261	00	00	321	23	LNX	381	00	0
	262	92	RTN	322	71	SBR	382	68	NOP
•	263	76	LBL	323	30	TAN	•383	56	DEL
∘	264	42	STO	324	61	GTO	∘384	42	STO
	265	65	×	325	38	SIN	385	03	03
	266	43	RCL	326	68	NOP	386	01	1
	267	00	00	•327	76	LBL	387	00	0
	268	95	=	∘328	24	CE	388	01	1
	269	69	OP	329	43	RCL	389	00	0
	270	02	02	330	04	04	390	01	1
•	271	92	RTN	331	71	SBR	•391	46	INS
∘	272	76	LBL	332	44	SUM	∘392	56	DEL
	273	44	SUM	333	92	RTN	393	68	NOP
	274	65	×	334	68	NOP	394	00	0
	275	43	RCL	•335	76	LBL	395	01	1
	276	00	00	∘336	30	TAN	396	68	NOP
	277	95	=	337	71	SBR	397	00	0
	278	68	NOP	338	29	CP	398	00	0
•	279	68	NOP	339	71	SBR	•399	56	DEL
∘	280	56	DEL	340	34	FX	∘400	42	STO
	281	69	OP	341	92	RTN	401	04	04
	282	03	03	342	68	NOP	402	25	CLR
	283	69	OP	•343	76	LBL	403	61	GTO
	284	05	05	∘344	34	FX	404	13	C
	285	69	OP	345	43	RCL	405	68	NOP
	286	00	00	346	01	01	406	56	DEL
•	287	92	RTN	347	71	SBR	•407	68	NOP
∘	288	76	LBL	348	44	SUM	408	00	0
	289	13	C	349	92	RTN			
	290	71	SBR	350	68	NOP			
	291	90	LST	•351	56	DEL			
	292	01	1	∘352	76	LBL			
	293	00	0	353	29	CP			
	294	42	STO	354	43	RCL			
•	295	11	11	355	03	03			
∘	296	56	DEL	356	71	SBR			
	297	71	SBR	357	42	STO	∘ PSP_A		
	298	22	INV	358	68	NOP	• PSP_E		
	299	76	LBL	•359	92	RTN			

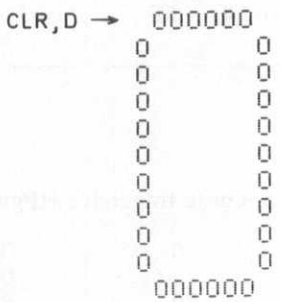


Bild 5: Null-Druck

Martin Ernst

Single Step für Basic-Programme auf dem TRS-80 Level II

Um den Nutzen meines TRS-80 weiter zu erhöhen, entwickelte ich folgendes kurzes Maschinenprogramm:

Nach dem Assemblieren und Speichern auf Kassette kann man das Programm mit dem SYSTEM-Befehl laden. Danach mit "/" und Enter starten.

Gibt man jetzt den Diskbefehl CMD ein, der ja im Level II nicht benutzt ist, so wird der Single-Step eingeschaltet. Bei jedem Betätigen der Leertaste wird ein BASIC-Befehl übersetzt. Auch beim LIST-Befehl ist der Single-Step aktiviert. Hierbei wird aber eine ganze BASIC-Zeile übersetzt.

Aufpassen muß man, da auch zum Ausdrucken des „READY" die Routine angesprungen wird. Wenn nun der Single-Step eingeschaltet ist, muß man immer, wenn ein „READY" zu schreiben ist (z.B. am Ende eines Listings oder nach Ausführung eines BASIC-Befehls im Direct Mode usw.), zusätzlich die Leertaste einmal betätigen.

Zusammen mit der Tracemöglichkeit der Befehle TRON/TROFF hat man nun eine einfache Möglichkeit, seine Programme auszutesten und so Fehler zu finden.

PS: Es kann vorkommen, daß nach dem Laden und Starten mit "/", Enter ein OM ERROR erscheint, sobald man etwas eingeben will und mit Enter abgeschlossen hat. Dies umgeht man am einfachsten dadurch, daß man nach dem Erscheinen des „READY" irgendeinen Buchstaben eintippt und dann Enter drückt. Dabei erscheint ein SN ERROR, aber alle Pointer werden wieder berichtigt.

```
        org     7FD0 H                  ;mem size 32720
start   ld      hl,single
        ld      (41C5 H),hl             ;änderung des vectors
        ld      hl,cmd
        ld      (4174 H),hl             ;änderung des cmd befehls
        jp      1A19 H                  ;zurück nach BASIC
cmd     ld      a,(flag)                ;holt flag zum umschalten
        cp      01 H                    ;single step eingeschaltet?
        jr      z,rset                  ;wenn ja, zurückschalten
        ld      a,C3 H                  ;erstes Byte des jump befehls
        ld      (41C4 H),a              ;single step ein!
        ld      a,1
        ld      (flag),a                ;flag setzen
        ret                             ;ende der umschaltroutine
rset    ld      a,C9 H                  ;return befehl
        ld      (41C4 H),a              ;single step aus!
        ld      (flag),a                ;flag = 01 setzen
        ret                             ;ende der zurückschaltroutine
single  push    hl                      ;routine warten auf leertaste
        push    bc
        push    de
        push    af                      ;alle register gesichert
wait    call    03E3 H                  ;tastaturabfrage
        cp      20 H                    ;leertaste gedrückt?
        jr      z,weiter                ;wenn ja, dann weiter
        jr      wait                    ;neue abfrage
weiter  pop     af
        pop     de
        pop     bc
        pop     hl                      ;register wiederherstellen
        ret                             ;ende der warteschleife
flag    defb    10 H
        end     start                   ;autostart
```

Heino Velder

BASIC-Load für CBM

Beschreibung

Jeder, der schon einmal versucht hat, zwei BASIC-Programme miteinander zu verknüpfen, wird schnell festgestellt haben, daß der CBM beim Laden stets das alte Programm überschreibt. Die einzige Möglichkeit, dennoch einen — wenn auch kleinen — Programmteil zu übernehmen, besteht über das Bildschirm-RAM.

Das Problem ist jedoch mit nur 74 Byte zu lösen, wobei Sie statt LOAD einfach SYS 826 eingeben. Sie können nun beliebig viele Programmteile verknüpfen, sofern der verbleibende Speicherraum ausreicht. Auch müssen Sie streng darauf achten, daß sich die Zeilennummern der betreffenden Programmteile nicht überschneiden.

Zum Programm im einzelnen:
Bei dem Programmteil von $033A-351 handelt es sich lediglich um eine Kopie der LOAD-Routine. Nachdem der CBM ein Programm (ggf. mit Programmnamen) gefunden hat, folgt die Anfangs- und Endadresse. Die Anfangsadresse liegt bei BASIC-Programmen stets bei $0400; die Endadresse hängt natürlich von der Programmlänge ab. Der nachfolgende Programmteil ermittelt nun die Länge des vorhandenen BASIC-Programms und addiert diesen Betrag zu der Anfangs- und Endadresse. Nun wird der Ladevorgang in der ROM-Routine fortgesetzt, wobei das Programm jedoch verschoben, das heißt nach dem schon vorhandenen BASIC-Programm geladen wird.

Dieses Maschinenprogramm kann ohne Veränderungen in jeden Speicherraum verschoben werden.

Mehrfaches BASIC-LOAD

```
033A:  20 67 F6    JSR  $F667   ← Tape-Puffer-Adresse setzen
033D:  20 3B F8    JSR  $F83B   ← ? 'PRESS PLAY ON TAPE# 1'
0340:  20 FF F3    JSR  $F3FF   ← ? 'SEARCHING'
0343:  A9 00       LDA  =$00    :
0345:  20 AE F5    JSR  $F5AE   ← Tapeheader suchen
0348:  D0 03       BNE  $034D   ← gefunden
034A:  4C 79 F5    JMP  $F579   ← ? 'FILE NOT FOUND ERROR'
034D:  E0 01       CPX  =$01    :
034F:  D0 F2       BNE  $0343   :
0351:  20 4D F6    JSR  $F64D   ← Anfangs-,Endadresse holen
0354:  D8          CLD          ┐
0355:  A2 01       LDX  =$01    |
0357:  A9 04       LDA  =$04    | Sucht nach der Adresse der
0359:  86 F7       STX  $F7     |
035B:  85 F8       STA  $F8     | letzten BASIC-Zeile
035D:  A0 00       LDY  =$00    |
035F:  B1 F7       LDA  ($F7),Y |
0361:  AA          TAX          | $F7, F8 = Anfangsadresse
0362:  C8          INY          |
0363:  B1 F7       LDA  ($F7),Y | für das BASIC-Programm
0365:  F0 02       BEQ  $0369   |
0367:  D0 F0       BNE  $0359   ┘
0369:  A5 F7       LDA  $F7     ┐
036B:  38          SEC          |
036C:  E9 01       SBC  =$01    | Adresse um eins
036E:  85 F7       STA  $F7     |
0370:  B0 02       BCS  $0374   | erniedrigen
0372:  C6 F8       DEC  $F8     ┘
0374:  18          CLC          ┐
0375:  A5 E5       LDA  $E5     |
0377:  65 F7       ADC  $F7     | Endadresse berechnen
0379:  85 E5       STA  $E5     |
037B:  A5 E6       LDA  $E6     | und in $E5, E6 ablegen
037D:  65 F8       ADC  $F8     |
037F:  85 E6       STA  $E6     ┘
0381:  4C C6 F3    JMP  $F3C6   ← LOAD fortsetzen
```

Ihre »Management-Wissen« Spezial-Bibliothek

Sieben Bände, die kurz, übersichtlich und leicht verständlich alles bringen, was man heute in Management-Spezial-Disziplinen wissen muß:

Ideenfindung

Eine höchst anregende und sofort umsetzbare praktische Einführung in tatsächlich alle bekannten und weniger bekannten Methoden der Ideenfindung. Dazu Innovationsprozesse im Unternehmen, Kreativitätsförderung und Organisation der Ideenfindung. 130 Seiten, 20,– DM

Methoden

30 kurzgefaßte Beiträge über Arbeitstechniken und Managementmethoden. Ein eigenes Kapitel über Rhetorik, Dialektik und Verhandlungstechnik sowie konkrete Anleitungen für Streßbewältigung und Selbstmanagement. Für alle Führungskräfte, die sich selbst besser managen wollen. 128 Seiten, 20,– DM

Mitarbeiterführung

Ein praxisbezogener Querschnitt durch alle wichtigen Führungsaufgaben. 36 kurzgefaßte Beiträge über Modelle moderner Führung, Führungsorganisation und Führungsmittel sowie Konfliktsituationen und Führungsfehler. Denkanstöße, die jedem Vorgesetzten helfen, sein Führungsverhalten zu verbessern. 130 Seiten, 20,– DM

Marketing

Eine praktische Einführung in marktbezogenes Denken und Handeln. Die wichtigsten Grundlagen und Strategien, Marktforschung in Eigenregie sowie Verkäuferverhalten und Verkaufsgespräch. Für jeden Betrieb, der Marketing nicht als theoretisches Schlagwort, sondern als Denk- und Geisteshaltung verstehen möchte. 116 Seiten, 20,– DM

Betriebswirtschaft

32 kurzgefaßte und fundierte Beiträge über alle wichtige Aufgaben und Probleme der praktischen Betriebswirtschaft. Von der Planung und Unternehmenssteuerung bis Finanzierung und Liquidität. Besonders für Unternehmer, die dafür keinen „Controller" haben. 146 Seiten, 24,– DM

Wirtschaftspolitik

Eine Einführung in die wichtigsten Aufgaben moderner Wirtschaftspolitik. 28 leichtverständliche Beiträge über Probleme der Weltwirtschaft und der deutschen Wirtschaft. Viele aktuelle Statistiken. Für alle, die mehr wissen wollen, als in der Zeitung steht. 128 Seiten, 20,– DM

Wirtschaftstheorie

Eine leicht verständliche und umfassende Einführung in die allgemeine Volkswirtschaftslehre. 28 kurzgefaßte Beiträge über die wichtigsten theoretischen Grundlagen, ohne die man heutiges Wirtschaftsgeschehen nicht verstehen kann. Volkswirtschaftliche Systeme, Modelle und Lehren, auch im geschichtlichen Rückblick. Für alle, die die tägliche Wirtschaftspolitik vor einem theoretischen Hintergrund verstehen wollen. 127 Seiten, 20,– DM

Alle Bände zu beziehen über

VOGEL-VERLAG

D-8700 Würzburg, Postfach 67 40
Abt. »Management-Wissen« (06)

An Vogel-Verlag, »Management-Wissen«, Abt. 06, Postf. 67 40, D-8700 Würzburg 1

Bestell-Coupon

Hiermit bestelle ich/wir

Ex. _____

_____

_____

Adresse

_____

_____

_____

_____

Datum/Unterschrift

✂ Bitte nennen Sie Ihre Wünsche auf nebenstehendem Bestellcoupon

Die Verknüpfung von Computern und Naturwissenschaften ist offensichtlich — das Spektrum der Programme riesengroß. Unsere Auswahl reicht von Taschenrechner-Numerik über die Berechnung des Pascalschen Dreiecks bis hin zum Fourierkoeffizienten.
Sie finden auf den nächsten Seiten folgende naturwissenschaftliche Anwendungsprogramme:
Taschenrechner-Numerik auf dem TI-59
Berechnung von Fourierkoeffizienten für Casio fx-502
Basic-Programm zur Berechnung von D/A-Wandlerfehlern
Mengenlehre auf TI-99
Wheatstone-Brücke
Sternzeit — Sternposition mit dem TI-59
TI-59 berechnet Elektronenkonfiguration der Elemente
Mondposition mit TI-58/59
Ewiger Kalender auf dem PC 1211
Berechnung des Pascalschen Dreiecks auf TI-59

Marcus von Cube

Taschenrechner-Numerik auf dem TI-59

Beschreibung

Soweit die Spiele und — in Ansätzen — die Textverarbeitung schon Einzug in die Taschenrechnerprogrammierung genommen haben, bleibt deren Hauptanwendung doch die Numerik — das Rechnen mit Zahlen. So sind Rechner wie HP-41 C und TI-59 mit weitaus mehr numerischen Möglichkeiten ausgestattet als jeder Basic-Interpreter. Die Grenzen liegen hauptsächlich im verfügbaren Speicherplatz und in der Rechengeschwindigkeit der Kleinen. Gerade hier also lohnt sich die Programmierung effizienter Algorithmen.

Ein typisches Problem ist zum einen die Lösung großer Systeme linearer Gleichungen — auf dem TI-59 im Rahmen der Möglichkeiten mit dem Bibliotheks-Programm zufriedenstellend gelöst — und zum anderen die Lösung einer einzigen, aber nichtlinearen Gleichung.

Die Gleichung
Linke Seite (x) = Rechte Seite (x),
wobei beide Seiten beliebige arithmetische Ausdrücke sein können, formuliert man am besten

- als Nullstellenproblem:
  Linke Seite (x) — Rechte Seite (x) $= f(x) \stackrel{!}{=} 0$
- oder als sogenanntes Fixpunktproblem:
  $x = g(x)$,
  wobei g sich aus f durch irgendeine Vorschrift so ergibt, daß für die Lösung x^* auch $f(x^*) = 0$ erfüllt ist.

Intervallhalbierung

Die Intervallhalbierung ist ein Verfahren zur Lösung von $f(x) = 0$.

Ausgehend von zwei Näherungen x_0, x_1, für die gelten muß

signum $(f(x_0))$ = — signum $(f(x_1))$,

berechnet man sich $\hat{x} = 0.5^* (x_0 + x_1)$ (Mitte des Intervalls) und prüft, in welchem Teilintervall, $(x_0, \hat{x})$ oder $(\hat{x}, x_1)$ die Funktion f ihr Vorzeichen wechselt. Dann werden die linke Intervallgrenze in x_0 und die rechte in x_1 umgetauft und das Verfahren beginnt von Neuem.
Der Abstand von $\hat{x}$ und der wahren Lösung x^* halbiert sich in jedem Schritt, in drei Schritten gewinnt man ungefähr eine (!) Dezimalstelle. Die Vorteile des Verfahrens sind seine Sicherheit — hat man einen Vorzeichenwechsel gefunden, findet man auch eine Nullstelle — und man hat immer eine Information über den Fehler, er ist kleiner als die halbe Intervallbreite. Aber das Verfahren ist langsam: Wer sich davon überzeugen möchte, probiere das mit dem Programm ML-08 von TI aus. Das Programm wurde mit

$x_0 = 1, \quad x_1 = 2, \quad \text{eps} = 10^{-5}$

und $f(x) = x^2 - 2$

aufgerufen und brauchte dafür 19 Iterationen.
Das gleiche Problem mit meinem Programm kam bei $x_0 = 2$ und sonst gleichem f und eps mit 3 Iterationen, das sind 6 Funktionsaufrufe, aus! Außerdem war die erreichte Genauigkeit um eine Stelle größer.

Die Regula falsi

Die „falsche Regel" (so die Übersetzung) nutzt mehr Informationen über die Funktion f aus, als das Intervallhalbierungsverfahren, das sich ja nur über das Vorzeichen von f an zwei Punkten eine neue Näherung beschaffte. x wird nicht mehr in die Intervallmitte gelegt, sondern nach Möglichkeit näher an x^*, indem man die Größe von $f(x_0)$ und $f(x_1)$ heranzieht und das gewichtete Mittel

$$\hat{x} = x_0 \cdot \frac{f(x_1)}{f(x_1) - f(x_0)} + x_1 \cdot \frac{f(x_0)}{f(x_0) - f(x_1)}$$

bildet und dann wie bei der Intervallhalbierung weiterverfährt.
Geometrisch ist $\hat{x}$ die Nullstelle der Geraden durch die Punkte $(x_0, f(x_0))$ und $(x_1, f(x_1))$.
Das Verfahren funktioniert sogar dann noch, wenn x^* nicht zwischen x_0 und x_1 liegt (Bild 1).

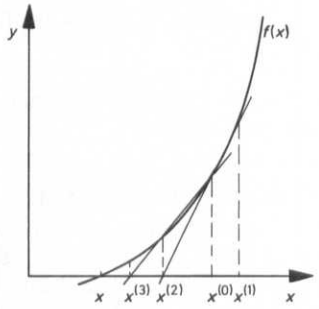


Bild 1

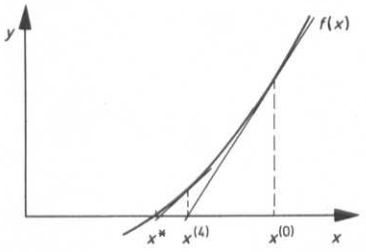


Bild 2

Ersetzt man im Beispiel aber den Sinus durch den Tangens, so „iteriert sich der Rechner tot", selbst wenn man mit $x(0) = 3{,}141592654$ (10 Stellen) startet.
Schuld an diesem Verhalten ist die Ableitung der Funktion g. Bild 2 stellt g für Beispiel 1 zusammen mit der Winkelhalbierenden $y = x$ dar. Der Schnittpunkt von g mit dieser Geraden ist die gesuchte Lösung x^*. Die gestrichelten Linien geben den Verlauf der Iteration wieder.
In Bild 3 ist die Situation für den Tangens dargestellt, Startwert für die Iteration war $x^{(0)} = 3$.

Da x_0 und x_1 im Prinzip ihre Rollen tauschen können, läßt sich so eine Iterationsvorschrift angeben:

$$x^{(n+1)} = x^{(n-1)} \cdot \frac{y^{(n)}}{y^{(n)} - y^{(n-1)}} + x^{(n)} \cdot \frac{y^{(n-1)}}{y^{(n-1)} - y^{(n)}}$$

mit $y^{(k)} = f(x^{(k)})$.

In dieser Form des Verfahrens verdreifacht sich die Anzahl der gültigen Stellen in zwei Iterationsschritten in etwa.

Allgemeine Verfahren

Um die Nullstelle von f(x) zu finden, kann man auch versuchen, eine entsprechende Fixpunktgleichung

$x = g(x)$

zu lösen, indem man einfach

$x^{(n+1)} = g(x^{(n)})$

immer wieder berechnet.
Der einfachste Weg zu solch einer Funktion g zu kommen, ist, auf beiden Seiten der Nullstellengleichung x zu addieren:
$f(x) = 0 \Leftrightarrow x + f(x) = x$
Beispiel:

$f(x) = \sin x$
$g(x) = x + \sin x$

Programm: (TI-59)

```
000   Rad
001   +
002   sin
003   =
004   pause
005   RST
```

Nach 5 Schritten erhält der Rechner aus der Näherung 1 exakt die Zahl π (auf 12 Stellen).

```
000  76 LBL
001  19 D'      Aufruf
002  61 GTO     von f (x)
003  01  01
004  72  72
005  76 LBL
006  12  B
007  50 I×I
008  94 +/-
009  22 INV
010  28 LOG
011  34 √X
012  42 STO     R1 = √eps
013  01  01
014  86 STF
015  01  01
016  33 X²
017  91 R/S
018  76 LBL
019  15  E      gehe in
020  61 GTO     LRN-Modus
021  01  01
022  71  71
023  76 LBL
024  14  D
025  42 STO
026  03  03
027  71 SBR
028  43 RCL     Ausgabe
029  53  (      von x
030  19 D'
031  54  )
032  32 X↑T
033  02  2
034  01  1
035  05  5
036  05  5
037  04  4
038  04  4
039  05  5      "F(X)"
040  06  6
041  69 OP
042  04  04
043  32 X↑T
044  69 OP
045  06  06
046  91 R/S
047  76 LBL
048  11  A
049  42 STO     R3 = x(o)
050  03  03
051  87 IFF     eps
052  01  01     bekannt?
053  85  +
054  04  4
055  94 +/-     √eps = 10⁻⁴
056  22 INV
057  28 LOG
058  42 STO
059  01  01
060  86 STF
061  01  01
062  76 LBL
063  85  +      Start der
064  53  (      Iteration
065  43 RCL
066  03  03
067  99 PRT
068  66 PAU     Zwischen-
069  50 I×I     ergebnis
070  32 X↑T
071  01  1
072  77  GE
073  77  GE
074  32 X↑T     Max ( x(k) ,1)
075  76 LBL
076  77  GE
077  65  ×
078  53  (
079  24 CE
080  55  ÷
081  07  7
082  22 INV
083  28 LOG
084  54  )
085  42 STO     R2 = h(k)
086  02  02
087  43 RCL
088  01  01
089  54  )
090  42 STO     R5 = Max (,,)
091  05  05      x  eps
092  53  (
093  43 RCL
094  02  02
095  85  +
096  43 RCL
097  03  03
098  54  )
099  42 STO
100  00  00
101  53  (
102  19 D'      f (x(k) h(k))
103  54  )
104  42 STO
105  04  04
```

(bitte umblättern)

Nichtlineares auf TI-59

```
106  43 RCL
107  03  03
108  42 STO
109  00  00
110  53  (
111  19 D'        f(x(k))
112  54  )
113  53  (
114  35 1/X
115  65  ×
116  43 RCL
117  04  04       f(x + h) / f(x) − 1
118  75  -
119  01  1
120  54  )
121  53  (
122  35 1/X
123  65  ×
124  43 RCL
125  02  02
126  94 +/-
127  54  )        Δ(n)
128  44 SUM
129  03  03       R3 = x(n+1)
130  50 I×I
131  32 X↕T
132  43 RCL
133  05  05
134  22 INV
135  77  GE       Ende der Schleife
136  85  +
137  76 LBL
138  43 RCL
139  04  4        "X"
140  04  4
141  00  0
142  00  0
143  69 DP
144  04  04
145  43 RCL
146  03  03
147  69 DP
148  06  06
149  92 RTN
150  76 LBL
151  13  C
152  24 CE
153  01  1
154  07  7        "EPS"
155  03  3
156  03  3
157  03  3
158  06  6
159  69 DP
160  04  04
161  43 RCL
162  01  01
163  33 X²
164  69 DP
165  06  06
166  91 R/S
167  76 LBL
168  10 E'
169  90 LST        List
170  90 LST
171  31 LRN        LRN
172  33 X²
173  75  -         Beispiel
174  04  4         f(x) = x² − 4
175  92 RTN
176  92 RTN
```

Solange die Steigung von g größer ist als die der Winkelhalbierenden, divergiert das Verfahren, es „läuft auseinander". Analoges gilt, falls g fällt statt steigt (Bilder 4 und 5).
Bedingung für die Anwendbarkeit, die Konvergenz, des Verfahrens ist also

$$|g'(x)| < 1$$

in einer genügend großen Umgebung von x^*, die Anfangsnäherung muß in diesen Bereich fallen, außerdem muß der Fixpunkt x^* natürlich existieren, dann fallen nämlich auch alle anderen $x^{(n)}$ in diesen Bereich der „Kontraktion" (die Näherungen „ziehen sich zusammen").
Die genaue Formulierung dieses wichtigen Satzes aus dem Gebiet der Funktionsanalysis, lautet eingeschränkt auf den Bereich der reellen Funktionen (Banachscher Fixpunktsatz):
Hat eine Funktion g in einem Intervall I überall eine Ableitung größer als -1 und kleiner als 1 und liegen die Werte der Funktion alle wieder in I, so besitzt die Fixpunktgleichung
$x = g(x)$
genau eine Lösung x^* und das Iterationsverfahren

$$x^{(n+1)} = g(x^{(n)})$$

strebt gegen diesen Fixpunkt x^*, falls $x^{(0)}$ nur in I liegt.

Zurück zu dem Problem, wie man so eine Funktion findet:

Das Newton-Verfahren

Legt man an die Funktion f, deren Nullstelle x^* gesucht ist, in einem Näherungspunkt die Tangente und berechnet deren Nullstelle als neue Näherung, dann erhält man das Verfahren von Newton (Bild 6).

Die Iterationsformel, die sich ergibt lautet

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$$

Die Funktion g hat demnach die Form

$$g(x) = x - \frac{f(x)}{f'(x)}$$

und die Ableitung

$$g'(x) = 1 - \frac{(f'(x))^2 - f(x)\,f''(x)}{(f'(x))^2}$$

Am Fixpunkt gilt sogar $g'(x^*) = 0$ und mithin auch $|g'(x)| < 1$ in einer Umgebung um die gesuchte Lösung. Die Bedingungen des Fixpunktsatzes sind also erfüllt, und die Näherungen $x^{(1)}$, $x^{(2)}$, ... streben wirklich gegen x^*, falls nur, und das ist das einzige Problem, das sich in der Praxis stellt, $x^{(0)}$ nahe genug an x^* liegt.

Anmerkung:
Wenn an der Nullstelle von f die Tangente an die Funktion nicht gerade waagerecht ist ($f'(x^*) = 0$), dann verdoppelt sich in jedem Schritt die Anzahl der gültigen Dezimalstellen in etwa.

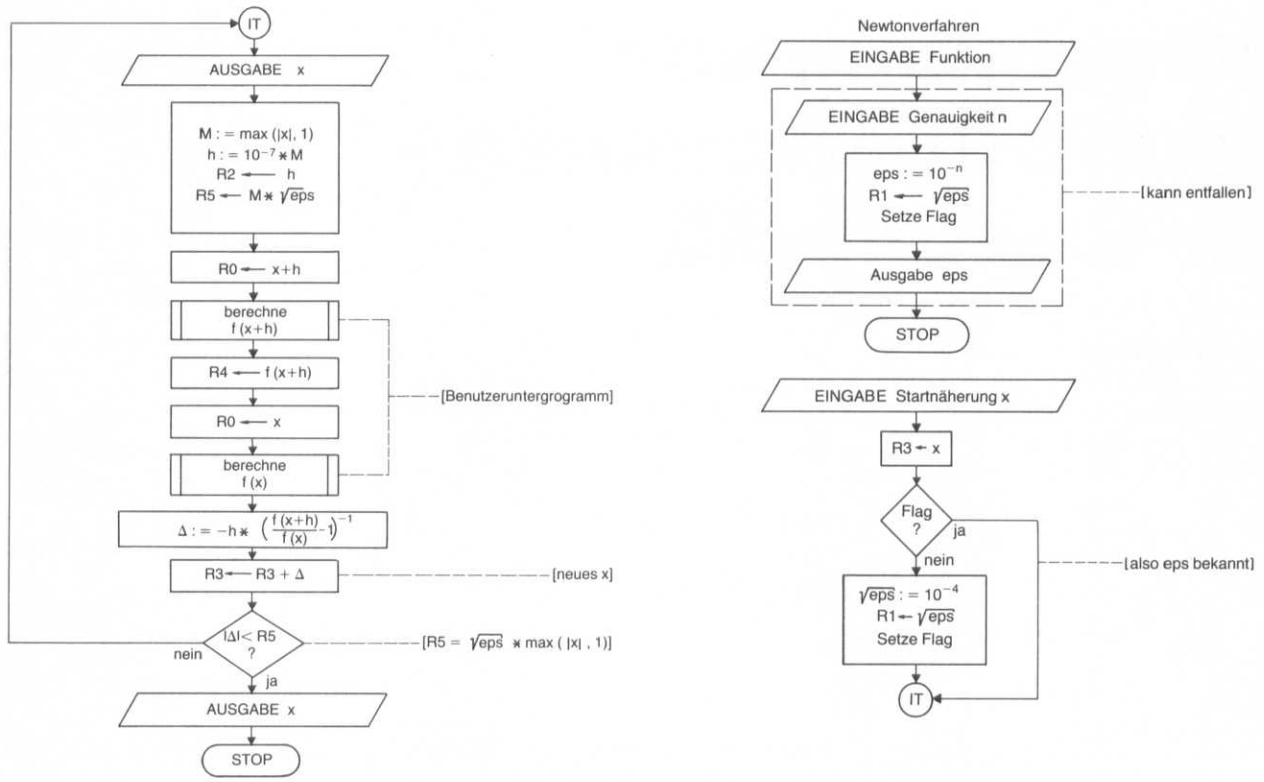
Das Programm

In der Newtonschen Formel kann man die Ableitung auch durch den Differenzenquotienten ersetzen:

$$f'(x) \approx \frac{f(x + h) - f(x)}{h}$$

Paßt man in jedem Schritt das h an, so geht das auf verschiedene Weise:

Mit $h^{(n)} = x^{(n)} - x^{(n-1)}$ erhält man wieder die Regula falsi.



In meinem Programm habe ich

$$h^{(n)} = 10^{-7} \cdot \max(|x^{(n)}|, 1)$$

gewählt
und die Newton-Formel
umgeformt zu

$$x^{(n+1)} = x^{(n)} - \Delta^{(n)} \quad \text{mit} \quad \Delta^{(n)} = h^{(n)} \cdot \left(\frac{f(x^{(n)} + h^{(n)})}{f(x^{(n)})} - 1 \right)^{-1}$$

Der Abbruch erfolgt, wenn

$$|\Delta^{(n)}| \leq \sqrt{\text{eps}} \cdot \max(|x^{(n)}|, 1)$$

eps ist also im Fall $|x^*| < 1$ eine absolute, sonst eine relative Fehlerschranke. Die Wurzel erklärt sich aus der Tatsache, daß das Verfahren immer Ergebnisse mit doppelt so vielen Stellen wie gefordert geliefert hat (Stellenverdoppelung in jedem Schritt!).

Die Anzahl N der gewünschten Stellen wird dem Programm mitgeteilt, für eps gilt dann:

$$\text{eps} = 10^{-N}$$

Nach Druck auf [E] verzweigt das Programm zur Speicherstelle 171 und geht in den LRN-Modus. Es empfiehlt sich, den Rest der Karte (bis 239) mit RTN zu füllen, dann braucht das nicht extra eingetippt zu werden.

Das Funktionsunterprogramm kann die Register ab R_{06} einschließlich sowie das t-Register und [=] frei verwenden. Das Programm braucht nicht mit (...) oder = abgeschlossen zu werden, denn der Aufruf erfolgt mit (D′).

Außer Flag 1 stehen alle Flags zur freien Verfügung. Das Argument steht beim Aufruf gleichzeitig in der Anzeige und in R_{00}.

Hat man ein längeres Funktionsunterprogramm auf Magnetkarte, so liest man diese in Block 2 ein und programmiert GTO 240 in Programmspeicherstelle 172 (nach Druck auf E).

Das Programm zeigt Zwischenergebnisse kurz an und protokolliert sie auf dem Drucker.

Abschließend das Beispiel aus Abschnitt „Intervallhalbierung“

$f(x) = x^2 - 2$

```
170  90 LST
171  31 LRN
172  33 X²
173  75 -
174  02 2
175  92 RTN
```

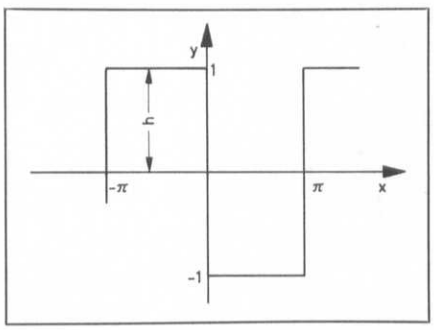
```
     0.00001  EPS
           2.
         1.5
1.416666667
1.414215669  X

           2.  √X
1.414213562
```

Michael Hüttmann

Berechnung von Fourierkoeffizienten für Casio fx-502

Beschreibung

Das vorliegende Programm ermöglicht die Berechnung von Fourierkoeffizienten. Näheres über den verwendeten Algorithmus ist in einschlägiger Literatur zu finden. Zur Demonstration sei aber nachfolgendes Beispiel aufgeführt:



$$f(x) = \frac{4\,h}{\pi}$$

$$(\sin x + 0 + \tfrac{1}{3}\sin 3x + \ldots)$$

für $-\pi \leq x \leq \pi$

Leider bietet dieses Programm infolge beschränkter Speicherkapazität keine allzugroße Genauigkeit bei den Ergebnissen. Je größer die Zahl der Stützstellen, um so genauer das Ergebnis.

Das Programm erlaubt eine Eingabe von maximal 12 Stützstellen:

Index	Stützstelle
1	1
2	1
3	1
4	1
5	1
6	1
7	−1
8	−1
9	−1
10	−1
11	−1
12	−1

Eingabemodus:

Angabe der nacheinander zu bedienenden Tasten:
PO
Anzahl der Stützstellen
EXE
Kennzahl
EXE
Ordnung des auszugebenden Polynoms
EXE
1. Stützstelle
EXE
⋮
n^{te} Stützstelle
EXE

Ausgabemodus:

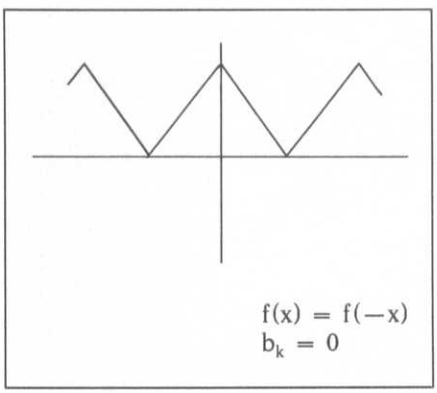
Nach jedem errechneten Koeffizienten ist die EXE-Taste zu drücken.
Zuerst Ausgabe von a_k, dann b_k
Blinkendes 0°0°0° bedeutet, daß entweder a_k oder b_k Null sind
Blinkendes 11°11°11° zeigt den Wechsel von a_k zu b_k an
Eine Fehlerverriegelung steht am Ende des Programms

Kennzahlen:

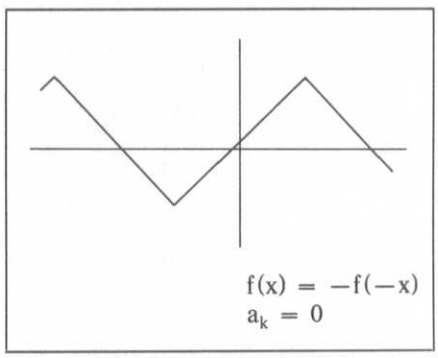
1 = gerade Funktion (Symmetrie 1. Art)
2 = unger. Funktionen (Symmetrie 2. Art)
3 = übrige Funktionen (Symmetrie 3.–5. Art)

Erläuterungen:

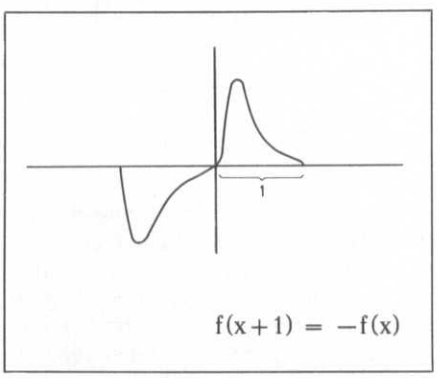
Beispiel einer Funktion für die Kennzahl 1 (gerade Funktion)



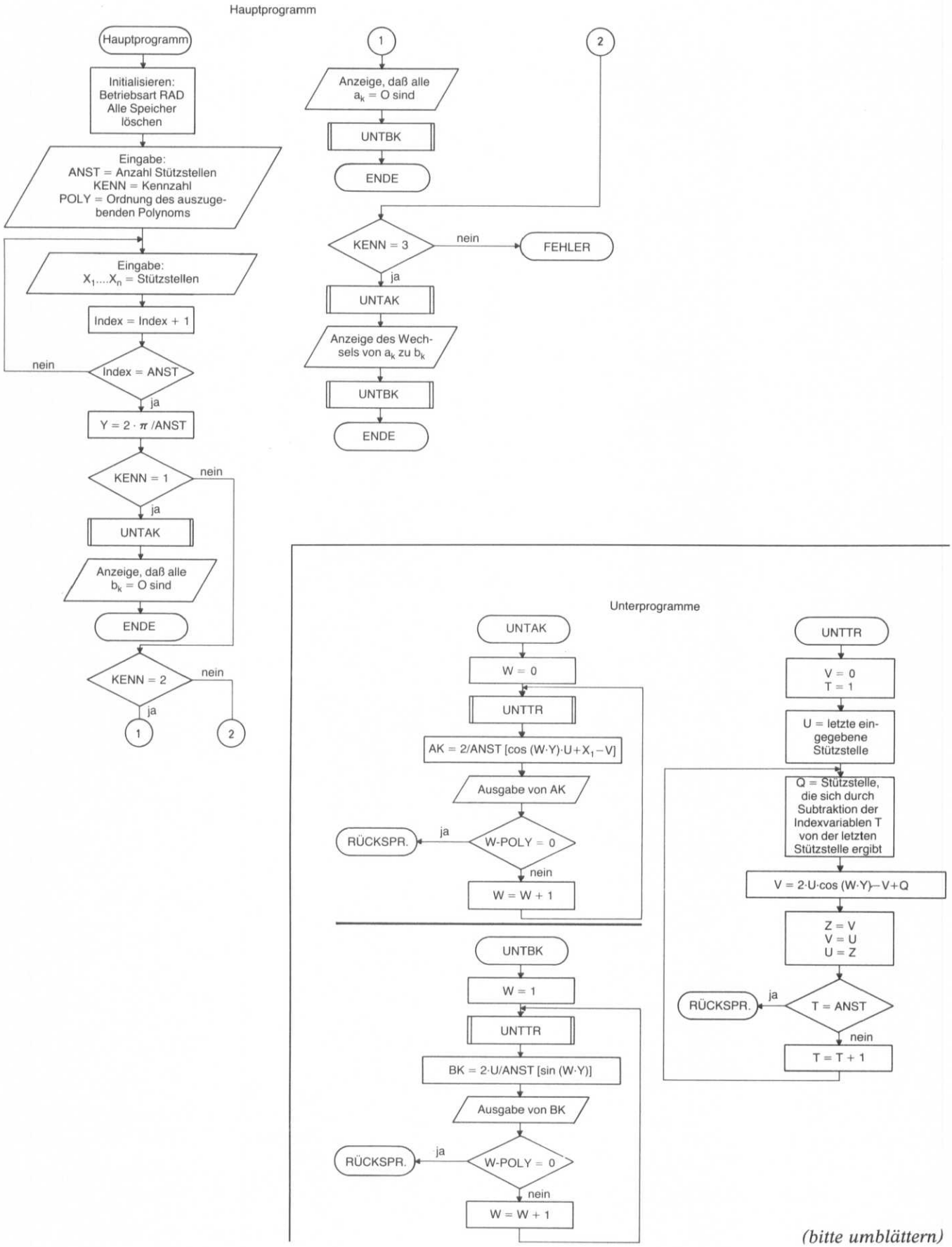
Kennzahl 2 (ungerade Funktion)



Kennzahl 3 (übrige Funktionen)



Die Ordnung des auszugebenden Polynoms entspricht der Anzahl von Koeffizienten, die errechnet werden sollen.



(bitte umblättern)

Berechnung von Fourierkoeffizienten für Casio fx-502

P0

```
 1   Mode 5
 2   INV MAC
 3   HLT
 4   Min F
 5   1
 6   Min 0
 7   3
 8   M+ F
 9   LBL 0
10   AC
11   HLT
12   INV IND
13   Min 0
14   INV ISZ
15   MR 0
16   INV x=F
17   GOTO 4
18   GOTO 0
19   LBL 4
20   3
21   M- F
22   π
23   x
24   2
25   +
26   MR F
27   =
28   Min 19
29   INV IND
30   GOTO 1
31   LBL 1
32   GSB P1
33   0
34   Min 18
35   GSB P4
36   GOTO 5
37   LBL 2
38   0
39   Min 18
40   GSB P4
41   GSB P2
42   GOTO 5
43   LBL 3
44   GSB P1
45-52 11.18639
53   Min 18
54   GSB P4
55   GSB P2
56   LBL 5
57   0
58   log
```

P1

```
59   0
60   Min 18
61   LBL 0
62   GSB P3
63   MR 18
64   x
65   MR 19
66   =
67   cos
68   x
69   MR 16
70   +
71   MR 3
72   -
73   MR  17
74   =
75   x
76   2
77   +
78   MR F
79   =
80   HLT
81   INV NOP
82   INV NOP
83   MR 18
84   -
85   MR 2
86   =
87   INV x=0
88   GOTO 1
89   1
90   M+ 18
91   GOTO 0
92   LBL 1
```

P2

```
 93   1
 94   Min 18
 95   LBL 0
 96   GSB P3
 97   MR 18
 98   x
 99   MR 19
100   =
101   sin
102   x
103   MR 16
104   x
105   2
106   +
107   MR F
108   =
109   HLT
110   INV NOP
111   INV NOP
112   MR 18
113   -
114   MR 2
115   =
116   INV x=0
117   GOTO 1
118   1
119   M+ 18
120   GOTO 0
121   LBL 1
```

P3

```
122   3
123   Min 15
124   0
125   Min 17
126   MR F
127   +
128   2
129   =
130   Min 0
131   INV IND
132   MR 0
133   Min 16
134   LBL 0
135   MR F
136   -
137   MR 15
138   +
139   4
140   =
141   Min 0
142   MR 18
143   x
144   MR 19
145   =
146   cos
147   x
148   2
149   x
150   MR 16
151   -
152   MR 17
153   +
154   INV IND
155   MR 0
156   =
157   Min 17
158   MR 16
159   X↔M 17
160   Min 16
161   MR 15
162   INV x=F
163   GOTO 1
164   1
165   M+ 15
166   GOTO 0
167   LBL 1
```

P4

```
168   3
169   Min 0
170   LBL 0
171   MR 18
172   INV ' "
173   INV PAUSE
174   INV DSZ
175   MR 0
176   INV x≥0
177   GOTO 0
```

Speicherinhalt:

MR 1 = Kennzahl
MR 2 = Ordnung des auszugebenden Polynoms
MR 3 - MR 14 = Stützstellen
MR F = Anzahl Stützstellen

Literatur:

H. Selder/Einführung in die numerische Mathematik für Ingenieure, Hanser Verlag;
Bronstein, Taschenbuch der Mathematik, Verlag Harri Deutsch, Thun und Frankfurt/Main, S. 668—670 (näheres über die Fourier-Approximation).

H. Zander

BASIC-Programm zur Berechnung von D/A-Wandler-Fehlern

Beschreibung

Der Gesamtfehler eines D/A-Wandlers wird im wesentlichen durch die Summe der drei folgenden Fehlerkomponenten bestimmt: Offset-Fehler (offset error), Skalenfaktor-Fehler (gain error) und Nichtlinearitäts-Fehler (linearity error). In Bild 1 sind die Zusammenhänge grafisch dargestellt. Offset-Fehler und Skalenfaktor-Fehler lassen sich bei den meisten Wandler-Moduln leicht auf Null abgleichen. Im Gegensatz dazu kann der Nichtlinearitäts-Fehler nicht abgeglichen werden.

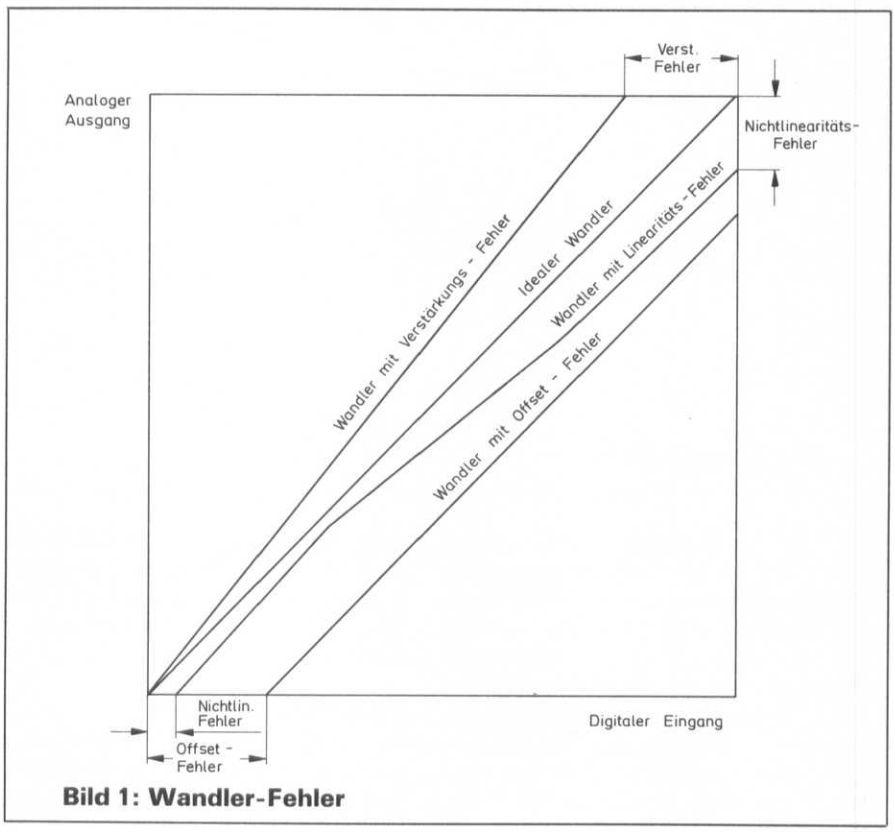


Bild 1: Wandler-Fehler

Offset-Fehler

Der Offset-Fehler ist die Abweichung der Übertragungsfunktion von der idealen Funktion gemessen im Nullpunkt. Wird der Offset-Fehler nicht auf Null abgeglichen, so tritt für jeden Punkt der Übertragungsfunktion ein absoluter Genauigkeits-Fehler auf. Die relative Genauigkeit wird nicht beeinflußt. Der Offset-Fehler wird in Prozent bezogen auf den vollen Skalenbereich (FSR) angegeben.

$$\text{Offset-Fehler}\ (\%)\ \frac{V_{FS_R} - V_{FS_i}}{V_{FSR_i}} \cdot 100$$

Skalenfaktor-Fehler

Der Skalenfaktor-Fehler ist die Abweichung der tatsächlichen Ausgangsfunktion von der idealen Übertragungsfunktion. Er wird in Prozent bezogen auf den theoretischen Wert angegeben. Der Offset-Fehler wird dabei nicht berücksichtigt.

$$\text{Skalenfaktor-Fehler}\ (\%) = \frac{\left[\sum_{i=1}^{n} V_{m_i} - nV_{FS_R}\right] - \left[V_{FSR_i} - \frac{V_{FSR_i}}{2^n}\right]}{V_{FSR_i}} \cdot 100$$

V_m = gemessener Wert

Nichtlinearitäts-Fehler

Die Nichtlinearität eines Wandlers ist die maximale Abweichung der realen Funktion von der idealen, bezogen auf die Endpunkte der Eingangs-/Ausgangsfunktion. Sie wird normalerweise in Bruchteilen eines LSB (least significant bit) oder in Prozent des Ausgangsspannungsbereiches (FSR) angegeben. Die Angabe von ±1/2 LSB für die Nichtlinearität bedeutet also, daß die Summe der negativen oder positiven Abweichungen der einzelnen Bit den Wert für 1/2 LSB nicht übersteigen darf.

Meßaufbau

Zum Testen des Wandlers wird eine stabile Gleichspannungsquelle (±0,1%) und ein Digital-Voltmeter benötigt. Die Genauigkeit des DVM sollte zehnmal besser sein, als die Linearität des Wandlers, z.B. ist bei einer Auflösung von 12 bit 1/2 LSB = 0,012% vom vollen Eingangsbereich. Für diesen Fall wird ein 5stelliges DVM (Genauigkeit ±0,001%) benötigt. Der Eingangskode des in Bild 2 dargestellten Wandlers ist „complementary offset binary", d.h., es ist ein Binär-Wand-

(bitte umblättern)

Basic-Programm zur Berechnung von D/A-Wandler-Fehlern

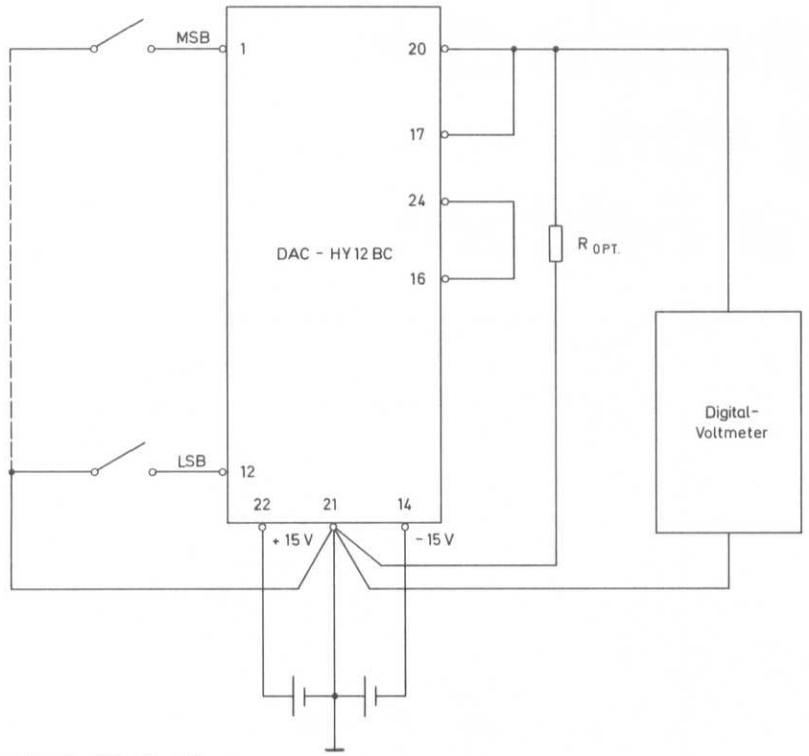


Bild 2: Meßaufbau

ler, der am Ausgang eine bipolare Spannung abgibt. Für positive Ausgangsspannungen schaltet ein 0-Pegel das entsprechende Bit ein. Um die für die Fehlerberechnung benötigten Werte zu erhalten, wird daher nacheinander, beginnend mit dem MSB jedes Bit auf 0-Potential gelegt und die Ausgangsspannung des Wandlers gemessen. Diese Werte werden in den Rechner eingegeben. Da der Rechner automatisch die ideale Übertragungsfunktion berücksichtigt, ist ein Abgleich des Wandlers vor der Messung nicht notwendig.

Programmablauf

Das Programm ist in Microsoft-Basic geschrieben und läuft ohne Modifikationen auf den CBM-Maschinen 2001, 3001 u.ä. In der vorliegenden Version ist der Anschluß eines Druckers vorgesehen. Da alle Ein- bzw. Ausgaben gleichzeitig auch auf dem Bildschirm erfolgen, kann nach einer Löschung der Zeilen 120, 130, 140, 160, 180, 200, 220, 270, 280, 290, 350, 410, 440, 530, 560, 590, 600 auf den Drucker verzichtet werden. Nach der Eingabe der gemessenen Werte werden als erstes die berechneten Bit-Fehler ausgegeben. Wegen der begrenzten Zeichenmenge bei der Ausgabe auf dem Bildschirm stoppt der Rechner nach Bit 8 und erst nach Eingabe von „y" erfolgt die Ausgabe der restlichen Bit (Zeile 460, 470, 480). Danach stoppt der Rechner ein zweites Mal die Ausgabe (Zeile 500, 510). Bei der erneuten Eingabe von „y" werden anschließend die Werte für den Offset-Fehler, den Verstärkungs-Fehler und den Nichtlinearitäts-Fehler ausgegeben.

Literaturhinweise

R. W. Jacobs: Specifying Testing Digital-to-Analog Converters Teledyne Philbrick AN 25 July 1976.

P. Prazak: Programmable Handheld Calculator Computes Digital-to-Analog Converter Errors, Burr Brown AN 94 November 1978,

Parameter, Fehlermechanismen und Standardisierung bei Datenerfassungsmoduln für Mikro- und Minicomputer,
Sonderdruck der Fa. Kontron November 1978.

```
100 REM WANDLERFEHLER ZANDER 1980 VERS 1.5 PLOTTMODE -WAFEPLO-
110 REM DATENEINGABE
120 OPEN1,4
130 PRINT#1,CHR$(31);"DIGITAL/ANALOG-UMSETZER FEHLER":PRINT#1,
140 PRINT#1,"EINGABE":PRINT#1,CHR$(30)
150 A$="FULL SCALE RANGE    (VOLT) ":PRINT A$,:INPUT F
160 PRINT#1,A$,F
170 A$="SCALE RANGE IDEAL   (VOLT) ":PRINT A$,:INPUT I
180 PRINT#1,A$,I:I=ABS(I)
190 A$="SCALE RANGE REAL    (VOLT) ":PRINT A$,:INPUT A
200 PRINT#1,A$,A:A=ABS(A)
210 A$="WELCHE AUFLOESUNG (IN BIT)":PRINT A$,:INPUT N
220 PRINT#1,A$,N,CHR$(10)
230 DIM B(N):Z=0:H$=" ":A$="BIT"
240 FOR L=1 TO N:L$=STR$(L):IF L(10 THEN L$=H$+L$
250 PRINT A$;L$;:INPUT X:X$=STR$(X)
260 PRINT#1,A$;L$,X;SPC(10-LEN(X$));"VOLT";
270 IF L=1 THEN PRINT#1,SPC(2);"(MSB)":GOTO 300
280 IF L=N THEN PRINT#1,SPC(2);"(LSB)"
290 PRINT#1,
```

```
 300 B(L)=X:Z=Z+X
 310 NEXT L
 320 REM BERECHNUNG
 330 E5=0:O=(I-A)/F*100:O=ABS(INT(O*10000)/10000)
 340 Z=ABS(Z):Y=F/2↑N:W=N*A:G=(W-Z-(F-Y))/F*100
 350 PRINT#1,CHR$(31);"AUSGABE":PRINT#1,CHR$(30)
 360 FOR M=1 TO N:L$=STR$(M)
 370 IF M<10 THEN L$=H$+L$
 380 E3=ABS(B(M)):E1=(A-E3)/(1+G/100)-(F/2↑M):E6=E1:E1=INT(E1*100000)/100
 390 A$="BIT":B$="BIT ERROR (VOLT)  ":C$="BIT ERROR (LBS)    ":D$="E-03"
 400 PRINT A$;L$;SPC(3);B$;E1;D$
 410 PRINT#1,A$;L$;SPC(3);B$;E1;D$
 420 E2=E6*2↑N/F:E7=E2:E2=INT(E2*10000)/10000
 430 PRINT SPC(9);C$;E2
 440 PRINT#1,SPC(9);C$;E2
 450 C=SGN(E7):IF C=1 THEN E5=E5+E7
 460 IF M<>8 THEN GOTO 490
 470 PRINT:PRINT" NAECHSTES BIT ? DANN 'Y'":PRINT
 480 GET Y$:IF Y$<>"Y" THEN GOTO 480
 490 NEXT M
 500 PRINT:PRINT" NAECHSTER WERT ? DANN 'Y'":PRINT
 510 GET Y$:IF Y$<>"Y" THEN GOTO 510
 520 A$="OFFSET ERROR (%)      ":PRINT A$;O
 530 PRINT#1,:PRINT#1,A$;O
 540 G=ABS(INT(G*10000)/10000)
 550 A$="GAIN ERROR   (%)      ":PRINT A$;G
 560 PRINT#1,A$;G
 570 E5=INT(E5*10000)/10000
 580 A$="LINEARITY ERROR (LSB) ":PRINT A$;E5
 590 PRINT#1,A$;E5
 600 CLOSE 1
READY.
```

↑ IN DEN ZEILEN 340 UND 380 STEHT FUER POTENZIEREN

Beispiel

EINGABE

```
FULL SCALE RANGE    (VOLT)  20
SCALE RANGE IDEAL   (VOLT)  10
SCALE RANGE REAL    (VOLT)  9.9973
WELCHE AUFLOESUNG (IN BIT) 12

BIT  1          -4E-04      VOLT  (MSB)
BIT  2           4.9985     VOLT
BIT  3           7.498      VOLT
BIT  4           8.7478     VOLT
BIT  5           9.3718     VOLT
BIT  6           9.6848     VOLT
BIT  7           9.841      VOLT
BIT  8           9.9193     VOLT
BIT  9           9.9585     VOLT
BIT 10           9.978      VOLT
BIT 11           9.9878     VOLT
BIT 12           9.9926     VOLT  (LSB)
```

AUSGABE

```
BIT   1   BIT ERROR (VOLT)  -.5 E-03
          BIT ERROR (LSB S) -.1007
BIT   2   BIT ERROR (VOLT)   .1 E-03
          BIT ERROR (LSB S)  .0213
BIT   3   BIT ERROR (VOLT)  -.05 E-03
          BIT ERROR (LSB S) -9.9E-03
BIT   4   BIT ERROR (VOLT)  -.18 E-03
          BIT ERROR (LSB S) -.0357
BIT   5   BIT ERROR (VOLT)   .56 E-03
          BIT ERROR (LSB S)  .1358
BIT   6   BIT ERROR (VOLT)   .08 E-03
          BIT ERROR (LSB S)  .0166
BIT   7   BIT ERROR (VOLT)   .09 E-03
          BIT ERROR (LSB S)  .0185
BIT   8   BIT ERROR (VOLT)  -.11 E-03
          BIT ERROR (LSB S) -.0215
BIT   9   BIT ERROR (VOLT)  -.26 E-03
          BIT ERROR (LSB S) -.0517
BIT  10   BIT ERROR (VOLT)  -.23 E-03
          BIT ERROR (LSB S) -.0464
BIT  11   BIT ERROR (VOLT)  -.27 E-03
          BIT ERROR (LSB S) -.0539
BIT  12   BIT ERROR (VOLT)  -.19 E-03
          BIT ERROR (LSB S) -.0372

OFFSET ERROR (%)        .0135
GAIN ERROR   (%)        .0261
LINEARITY ERROR (LSB)   .1924
```

Horst Treisch

Mengenlehre auf TI-99

Da der TI-99 mit seinem BASIC-Wortschatz keine logischen Entscheidungen im Sinne der Boolschen Algebra fällen kann, AND, OR und NOT sind nicht implementiert, muß ihm mit Dualzahlen und „normaler" Arithmethik auf die Sprünge geholfen werden. So kann auch der kassettenspielstarke TI für ein Mengentraining der schulpflichtigen Kinder oder für lernsüchtige Oldies fit gemacht werden.
Logische Operationen „AND" „OR" „NOT" mit dem TI-99 (Sprache: TI-BASIC) **ausgeführt mit arithmetischen Operatoren**
„AND" .....

„OR" .....

„NOT" .....

IF (A*B) THEN

IF (A+B) THEN

```
100  IF P=1 THEN 130
                    (-1) bei TI-BASIC
110  P=1            (-1) bei TI-BASIC
120  GOTO 140
130  P=0
140  ...
```

A B P sind Boolsche Variable (0 oder 1)

Dual-Zahlen und natürliche Zahlen
Es gibt **keinen** Unterschied. Dual-Zahlen sind lediglich ein Sonderfall der natürlichen Zahlen. Es gelten die Regeln der Algebra.

Beweis: Am Beispiel eines **zwei**stelligen **Zählers**

```
100  FOR A = 0 TO 1
110  FOR B = 0 TO 1
120  GOTO 160
130  NEXT B
140  NEXT A
150  STOP
160  REM LOGIK AND/OR
170  IF (A*B) THEN 180 ELSE 130
180  PRINT A; B
190  GOTO 130
```

(Anmerkung: Bewußt wird hier aus der Schleife gesprungen, um beliebige Logik-Operationen durchführen zu können. Ein Einsprung in eine eröffnete Schleife ist jederzeit möglich.)

Beispiel für **oder** „OR"
170 IF (A+B) THEN 180 ELSE 130

Für die Logik-Operationen gilt nur die Entscheidung: Ist das Resultat **Null oder nicht Null.**

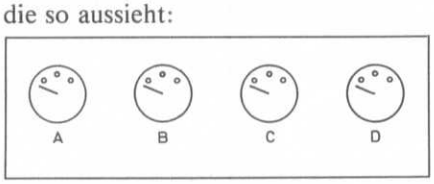
Aus dem zweistelligen Zähler wird ein Dezimalzähler oder ein anderer Zähler in einem beliebigen Zahlensystem, indem die obere Grenze entsprechend geändert wird (nicht bis 1 sondern bis 9 usw.).

Das nachfolgende Beispiel „Maschinenlogik" unterstreicht die Richtigkeit meiner Behauptung durch reichhaltige Anwendung der arithmetischen Operatoren * und + bei Logik-Verknüpfungen.

Zweistein Nr. 39 „Maschinenlogik"

Aufgabe:

Möchten Sie gerne verhindern, daß Ihre kleinen Kinder selbständig den Fernseher einschalten? Dann verrate ich Ihnen einen Trick. Sie lassen sich vom Elektriker eine Schalttafel bauen, die so aussieht:



Diese Schalttafel soll mit dem Fernseher so verkoppelt werden, daß er nur dann eingeschaltet wird, wenn die vier Knöpfe in eine ganz bestimmte Stellung gebracht worden sind. Wie, das lehrt die Gebrauchsanweisung:

1. Jeder Knopf hat vier verschiedene Einstellungsmöglichkeiten.
2. Keine der zwei Knöpfe dürfen auf der gleichen Zahl stehen.
3. Falls weder Knopf A auf der 1 noch Knopf B auf der 2 richtig stehen, muß entweder Knopf C oder Knopf D auf der 4 stehen.
4. Mindestens eine der folgenden vier Knopfstellungen ist richtig: A auf 3, B auf 1, C auf 2, D auf 3.
5. Stehen weder A noch B auf der Stellung 4 richtig, dann muß mindestens eine der beiden folgenden Stellungen richtig sein: C auf 3, D auf 2.
6. Falls weder Knopf D auf der 1 noch Knopf B auf der 3 richtig stehen, muß der Knopf A oder Knopf C auf der 2 stehen.
7. Wenn D auf 3 steht, dann darf C nicht auf 2 stehen.

Nach dieser Gebrauchsanweisung bringt man die vier Knöpfe in ihre einzig richtigen Stellungen, und der Fernseher ist angeschaltet. So einfach ist das.

Wie müssen die vier Knöpfe eingestellt werden?

```
100 REM MASCHINENLOGIK
    ZWEISTEIN NR. 39
120 PRINT "LOGISCHE OPERATIONEN MIT ARITHMETISCHEN
    OPERATOREN + * "
130 PRINT "SICHERHEITSSCHALTUNG
    FERNSEHER 4 KNÖPFE"::
140 FOR A=1 TO 4 |
150 FOR B=1 TO 4 |                    ZAHLER
160 FOR C=1 TO 4 |
170 FOR D=1 TO 4 |
180 GOTO 240
190 NEXT D
```

```
200 NEXT C
210 NEXT B
220 NEXT A
230 STOP
240 IF (A=B) THEN 210
250 IF (A=C)+(B=C) THEN 200
260 IF (A=D)+(B=D)+(C=D)
    THEN 190
270 REM LOGIK MIT ARITHMETISCHEN
    OPERATOREN * +
280 IF (A<>1)*(B<>2)*((C=4)+(D=4))=0
    THEN 190
290 IF (A=3)+(B=1)+(C=2)+(D=3)=0 THEN 190
300 IF ((A<>4)+(B<>4))*((C=3)
    +(D=2))=0 THEN 190
310 IF (D<>1)*(B<>3)*((A=2)+(C=2))=0
    THEN 190
320 IF (D=3)*(C=2) THEN 190
330 PRINT A;B;C;D
340 PRINT :: "AUF DIESE ZAHLEN SIND DIE
    SCHALTER
    A B C D  EINZUSTELLEN"
350 GOTO 190
```

PERMUTATIONS-LOGIK +

Als Resultat erscheint: 2 1 3 4

Auf diese Zahlen sind die Schalter
A B C D einzustellen.

Kommentar zum Programm „Maschinenlogikzweistein Nr. 39 8—80

140···170 Vierstelliger Zähler — je 4 Werte (Ziffer 1)

240···260 Permutationslogik (vgl. Ziffer 2 der Aufgabe)

280···320 Logik (vgl. Ziffer 3 ...7 der Aufgabe)

```
100 REM PERMUTATION ZEICHEN AUG 80  ZEICHEN-
110 CALL CLEAR
120 PRINT          "PERMUTATION
    KETTE 6 ZEICHEN LANG"::
130 DIM W$(6) (kann auch größer sein)
140 INPUT "ZEICHENKETTE S$ : ":S$
150 N=LEN(S$)
160 FOR X=1 TO N
170 W$(X)=SEG$(S$, X, 1)
180 NEXT X
190 FOR A=1 TO N
200 FOR B=1 TO N
210 FOR C=1 TO N
220 FOR D=1 TO N
230 FOR E=1 TO N
240 FOR F=1 TO N
250 GOTO 330
260 NEXT F
270 NEXT E
280 NEXT D
290 NEXT C
300 NEXT B
310 NEXT A
320 STOP
330 IF (A=B) THEN 300
340 IF (A=C)+(B=C) THEN 290
350 IF (A=D)+(B=D)+(C=D) THEN 280
360 IF (A=E)+(B=E)+(C=E)+(D=E) THEN 270
370 IF (A=F)+(B=F)+(C=F)+(D=F)+(E=F)
    THEN 260
380 PRINT W$(A); W$ (B); W$(C); W$(D); W$(E);
    W$(F)
390 GOTO 260
```

EINGABE: Z.B.
123456 ODER ODER (#=3A*
ABCDEF

```
100 REM PERMUTATION 2 ZEICHEN AUG 80
110 CALL CLEAR
120 PRINT "PERMUTATION 2 ZEICHEN"::
130 INPUT "STRING 2 ZEICHEN S$ :":S$
140 N=LEN(S$)
150 IF N<>2 THEN 160 ELSE 180
160 PRINT "TRY AGAIN"::
170 GOTO 130
180 FOR X=1 TO 2
190 W$(X-1)=SEG$(S$,X,1)
200 NEXT X
205 INPUT "WIE OFT DAS 2. ZEICHEN? ":V
210 REM DUALZAEHLER
220 FOR A=0 TO 1
230 FOR B=0 TO 1
240 FOR C=0 TO 1
250 FOR D=0 TO 1
260 FOR E=0 TO 1
270 FOR F=0 TO 1
280 FOR G=0 TO 1
290 FOR H=0 TO 1
300 FOR I=0 TO 1
310 GOTO 420
320 NEXT I
330 NEXT H
340 NEXT G
350 NEXT F
360 NEXT E
370 NEXT D
380 NEXT C
390 NEXT B
400 NEXT A
410 STOP
420 REM LOGIK-TESTS BINAER-VARIABLE ABCDEFGHI
430 IF (A+B)>V THEN 390
440 IF (A+B+C)>V THEN 380
450 IF (A+B+C+D)>V THEN 370
460 IF (A+B+C+D+E)>V THEN 360
470 IF (A+B+C+D+E+F)>V THEN 350
480 IF (A+B+C+D+E+F+G)>V THEN 340
490 IF (A+B+C+D+E+F+G+H)>V THEN 330
500 IF (A+B+C+D+E+F+G+H+I)<>V THEN 320
510 PRINT W$(A);W$(B);W$(C);W$(D);W$(E);
    W$(F);W$(G);W$(H);W$(I)
520 GOTO 320
```

Z.B. EINGABE AT/1 ODER 01/3 USF Z.B. "*"/3

Wilhelm-Rüdiger Haberditz

Wheatstone-Brücke

Beschreibung

In der Elektronik, der Elektrotechnik und in der Meß- und Regeltechnik usw. kommen Brückenschaltungen nach Wheatstone — siehe Bild 1 — sowohl in klassischer als auch in modifizierter Form relativ häufig vor. Wenn auch die Ermittlung der Kenndaten einer unabgeglichenen Brücke als verhältnismäßig einfach gelten kann, so ist sie doch meist etwas umständlich und zeitaufwendig.

Einfacher und rationeller geht's per Programm. Gut geeignet dafür ist der mittlerweile weitgestreute TI-59 mit Thermodrucker PC-100 C. Das lediglich auf einer Magnetbandkarte konservierte und allgemeingültige Programm — siehe Listing — belegt 400 Programmspeicherstellen (PSS) und 10 Register (R60···R69) zur Speicherung der Symbol-, Dimensions- und Textdaten. Bei der erstmaligen Eingabe der Befehle und Daten ist zu beachten, daß die DSZ 57-, DSZ 59- bzw. HIRn-Befehle durch Überschreiben entstehen, und die Register R60···R63 doppelt, d.h. mit dem alphanumerischen Kode für Symbol **und** Dimension belegt sind (4 + 8 Ziffern). Da vom Eingabeformat bekanntlich nur max. 10 Ziffern parallel verarbeitet werden, ist hierbei gesplittet einzugeben. Beispiel: R61 wird zunächst mit 4164 („U=") über STO 61 geladen. Anschließend wird 0.42322737 („VOLT") durch SUM 61 dazu addiert, so daß letztlich der 12stellige Kode 4164.42322737 in R61 steht. Der Programmablauf ist aus dem schematisierten Flußdiagramm ersichtlich. Die vorzeichenrichtige Netzwerkberechnung basiert auf den bekannten Gesetzen von G. R. Kirchhoff und G. S. Ohm. Entsprechend den geltenden Regeln sind in Bild 1 die Spannungszählpfeile identisch mit denen der Ströme eingetragen. Da im Diagonalzweig (b/c) die Richtung undefiniert ist, wurde der U- bzw. J-Pfeil willkürlich festgelegt. Entspricht die Richtungswahl nicht dem tatsächlichen Verlauf, so ergeben sich für J_5 und U_5 negative Lösungen, d.h., die Fluß- bzw. Potentialrichtung ist invers zu Bild 1. Wird die Polarität der Speisespannung (−U) geändert, so gilt dies auch für die übrigen Ströme und Spannungen.

Die praktische Anwendung des Programms ist denkbar einfach. Nach dem Einlesen der beiden Kartenseiten — normale Speicherbereichsverteilung SBV:479.59 — erfolgt die Initialisierung durch Drücken der Taste „A", wobei u.a. auch die im Ablauf vorgesehene SBV:399.69 automatisch eingestellt wird. Nach dem Ausdruck „EINGABE" werden die Widerstandswerte für (R_1, R_2, R_3, R_4, $R_5 > 0$) = Ohm und die Speisespannung (U) = Volt im Dialogbetrieb eingetastet und mit „R/S" eingegeben. Unzulässige Eingaben (R_1, R_2··· oder $R_5 = 0$) liefern den Ausdruck „0?"; eine neue Eingabe wird automatisch vorbereitet. R_1···R_5 mit negativem Vorzeichen werden vom Programm als |x| interpretiert. Für die anschließende Berechnung und den Textausdruck „WHEATSTONE-BRUEKKE" einschließlich der 24 Daten für „GESAMT" und „PFAD 1···PFAD 5", gekennzeichnet mit Symbol („R=", „U=", „J=", „P=") und Dimension („OHM", „VOLT", „AMP.", „WATT") — siehe Beispiele —, werden etwa 1,75 Minuten benötigt. Bei U=0 Volt wird automatisch nur der Netzwerk-Gesamtwiderstand R (a/d) ausgedruckt.

Ausgabe-Adressenschema für Text, Daten, Symbole und Dimensionen
Anfangsbedingungen: Schleifenzähler R57=6; Datenvoradresse R58=−6; Symbol- und Dimensionsadresse R59=60

Durchlauf →			1 (R57=5)	2 (R57=4)	3 (R57=3)	4 (R57=2)	5 (R57=1)	6 (R57=0)
Text →			„GESAMT"	R68+1= „PFAD 1"	+1=„PFAD 2"	+1=„PFAD 3"	+1=„PFAD 4"	+1=„PFAD 5"
R59	Symbol	Dimension	R58	R58	R58	R58	R58	R58
60	"R="	"OHM"	−6+ 6= 0	−5+ 6= 1	−4+ 6= 2	−3+ 6= 3	−2+ 6= 4	−1+ 6= 5
61	"U="	"VOLT"	0+ 6= 6	1+ 6= 7	2+ 6= 8	3+ 6= 9	4+ 6=10	5+ 6=11
62	"J="	"AMP."	6+ 6=12	7+ 6=13	8+ 6=14	9+ 6=15	10+ 6=16	11+ 6=17
63	"P="	"WATT"	12+ 6=18	13+ 6=19	14+ 6=20	15+ 6=21	16+ 6=22	17+ 6=23
Übertrag →			18−23=−5	19−23=−4	20−23=−3	21−23=−2	22−23=−1	23−23= 0

Da in den praktischen Schaltungen die Kenndaten vielfach in den Dimensionen kΩ, Ω, mV, V, nA, µA, mA, A, µW, mW, W usw. auftreten, wurde für die Ausgabe das sogenannte „technische Format" gewählt und auf 4 Kommastellen begrenzt. Damit ist bei gleichzeitig guter Lesbarkeit eine meist ausreichende Stellenzahl der Daten sichergestellt. Abhängig von der Anwendung läßt sich das Format (ENG, FIX 4) in PSS 030/031/032 leicht variieren.
Per Programm wird auch erkannt, ob Betrieb mit oder ohne Drucker vorliegt. Es ist keine Umstellung erforderlich! Der serielle Datenabruf erfolgt hierbei jeweils mit „R/S". Darüber hinaus sind nach jedem Ablauf alle R-, U-, J- und P-Werte ungerundet gespeichert, so daß weitere Berechnungen möglich sind (Abruf über „RCLnn").

Belegung der Datenregister

Widerstand	R:R00; R_1:R01; R_2:R02, R_3:R03; R_4:R04; R_5:R05
Spannung	U:R06; U_1:R07; U_2:R08; U_3:R09; U_4:R10; U_5:R11
Strom	J:R12; J_1:R13; J_2:R14; R_3:R15; J_4:R16; J_5:R17
Leistung	P:R18; P_1:R19; P_2:R20; P_3:R21; P_4:R22; P_5:R23

Kartenseiten 1u.2 einlesen;Programmvorbereitung mit Taste "A";Ausdruck:"EINGABE";[R1,R2,R3,R4,R5>0]=Ohm und [U]=Volt jeweils eintasten und mit "R/S" eingeben.Ausdruck:" WHEATSTONE-BRUECKE ". Daten ("R,U,J,P") für "GESAMT" und "PFAD 1...PFAD 5" ("OHM,VOLT,AMP.,WATT"). Bei R1...R5=0 erfolgt Ausdruck "0?"; eine neue Eingabe wird automatisch vorbereitet. Bei U=0 wird nur der Gesamtwiderstand (R) ausgedruckt.

PSS	Code	Befehl	PSS	Code	Befehl	PSS	Code	Befehl	PSS	Code	Befehl	PSS	Code	Befehl
000	61	GTO	080	42	STO	160	42	STO	240	06	06	320	02	02
001	00	●0	081	00	00	161	06	06	241	95	=	321	43	RCL
002	86	86	082	07	7	162	42	STO	242	49	PRD	322	66	66
003	76	LBL	083	69	DP	163	07	07	243	09	09	323	69	DP
004	16	A'	084	17	17	164	42	STO	244	43	RCL	324	03	03
005	73	RC*	085	81	RST	165	08	08	245	09	09	325	43	RCL
006	59	59	086	29	CP	166	42	STO	246	44	SUM	326	67	67
007	59	INT	087	01	1	167	11	11	247	10	10	327	69	DP
008	52	EE	088	07	7	168	42	STO	248	94	+/-	328	04	04
009	04	4	089	02	2	169	18	18	249	44	SUM	329	69	DP
010	22	INV	090	04	4	170	17	B'	250	07	07	330	07	07
011	52	EE	091	69	DP	171	43	RCL	251	44	SUM	331	69	DP
012	69	DP	092	01	01	172	03	03	252	08	08	332	19	19
013	01	01	093	03	3	173	55	÷	253	01	1	333	18	C'
014	18	C'	094	01	1	174	53	(	254	09	9	334	43	RCL
015	73	RC*	095	02	2	175	24	CE	255	42	STO	335	68	68
016	59	59	096	02	2	176	85	+	256	55	55	336	82	HIR
017	22	INV	097	01	1	177	43	RCL	257	01	1	337	03	•3
018	59	INT	098	03	3	178	01	01	258	03	3	338	06	6
019	52	EE	099	01	1	179	54	)	259	42	STO	339	42	STO
020	08	8	100	04	4	180	82	HIR	260	56	56	340	57	57
021	22	INV	101	01	1	181	07	•7	261	07	7	341	94	+/-
022	52	EE	102	07	7	182	95	=	262	42	STO	342	42	STO
023	69	DP	103	69	DP	183	42	STO	263	57	57	343	58	58
024	04	04	104	02	02	184	09	09	264	01	1	344	17	B'
025	06	6	105	18	C'	185	75	-	265	42	STO	345	06	6
026	44	SUM	106	17	B'	186	43	RCL	266	58	58	346	00	0
027	58	58	107	03	3	187	04	04	267	05	5	347	42	STO
028	73	RC*	108	05	5	188	55	÷	268	42	STO	348	59	59
029	58	58	109	08	8	189	53	(	269	59	59	349	87	IFF
030	57	ENG	110	02	2	190	24	CE	270	73	RC*	350	01	•1
031	58	FIX	111	06	6	191	85	+	271	57	57	351	03	•3
032	04	•4	112	04	4	192	43	RCL	272	55	÷	352	61	61
033	87	IFF	113	82	HIR	193	02	02	273	73	RC*	353	01	1
034	07	•7	114	04	•4	194	54	)	274	58	58	354	82	HIR
035	00	•0	115	05	5	195	82	HIR	275	95	=	355	33	33
036	38	38	116	42	STO	196	08	•8	276	72	ST*	356	82	HIR
037	91	R/S	117	59	59	197	95	=	277	56	56	357	13	13
038	69	DP	118	82	HIR	198	55	÷	278	33	X^2	358	61	GTO
039	06	06	119	14	14	199	53	(	279	65	×	359	03	•3
040	22	INV	120	69	DP	200	43	RCL	280	73	RC*	360	67	67
041	57	ENG	121	01	01	201	01	01	281	58	58	361	02	2
042	22	INV	122	18	C'	202	65	×	282	95	=	362	02	2
043	58	FIX	123	69	DP	203	43	RCL	283	72	ST*	363	69	DP
044	69	DP	124	20	20	204	03	03	284	55	55	364	01	01
045	00	00	125	91	R/S	205	55	÷	285	01	1	365	43	RCL
046	01	1	126	50	I×I	206	82	HIR	286	44	SUM	366	69	69
047	44	SUM	127	67	EQ	207	17	17	287	55	55	367	69	DP
048	59	59	128	03	•3	208	85	+	288	44	SUM	368	02	02
049	92	RTN	129	93	93	209	43	RCL	289	56	56	369	18	C'
050	76	LBL	130	99	PRT	210	02	02	290	44	SUM	370	17	B'
051	17	B'	131	72	ST*	211	65	×	291	57	57	371	16	A'
052	05	5	132	00	00	212	43	RCL	292	44	SUM	372	87	IFF
053	01	1	133	02	2	213	04	04	293	58	58	373	02	•2
054	05	5	134	22	INV	214	55	÷	294	97	DSZ	374	03	•3
055	55	÷	135	28	LOG	215	82	HIR	295	59	59	375	91	91
056	09	9	136	82	HIR	216	18	18	296	02	•2	376	16	A'
057	09	9	137	34	34	217	85	+	297	70	70	377	16	A'
058	95	=	138	97	DSZ	218	43	RCL	298	43	RCL	378	16	A'
059	82	HIR	139	59	59	219	05	05	299	13	13	379	02	2
060	05	•5	140	01	•1	220	49	PRD	300	85	+	380	03	3
061	82	HIR	141	18	18	221	11	11	301	43	RCL	381	22	INV
062	06	•6	142	04	4	222	95	=	302	14	14	382	44	SUM
063	82	HIR	143	01	1	223	49	PRD	303	95	=	383	58	58
064	07	•7	144	06	6	224	11	11	304	22	INV	384	22	INV
065	82	HIR	145	04	4	225	43	RCL	305	49	PRD	385	86	STF
066	08	•8	146	69	DP	226	11	11	306	00	00	386	01	•1
067	76	LBL	147	01	01	227	44	SUM	307	42	STO	387	97	DSZ
068	18	C'	148	18	C'	228	08	08	308	12	12	388	57	57
069	69	DP	149	91	R/S	229	94	+/-	309	49	PRD	389	03	•3
070	05	05	150	99	PRT	230	42	STO	310	18	18	390	44	44
071	69	DP	151	22	INV	231	10	10	311	86	STF	391	17	B'
072	00	00	152	67	EQ	232	55	÷	312	01	•1	392	91	R/S
073	25	CLR	153	01	•1	233	43	RCL	313	43	RCL	393	65	×
074	92	RTN	154	58	58	234	05	05	314	64	64	394	55	÷
075	76	LBL	155	86	STF	235	65	×	315	69	DP	395	99	PRT
076	11	A	156	02	•2	236	43	RCL	316	01	01	396	25	CLR
077	25	CLR	157	01	1	237	01	01	317	43	RCL	397	17	B'
078	98	ADV	158	42	STO	238	85	+	318	65	65	398	61	GTO
079	17	B'	159	00	00	239	43	RCL	319	69	DP	399	11	A

Registerbelegung

ALPHA-CODE	TEXT	NR
3564.32233000	R=OHM	60
4164.42322737	U=VOLT	61
2564.13303340	J=AMP.	62
3364.43133737	P=WATT	63
5143231713.	*WHEA	64
3736373231.	TSTON	65
1720143541.	E-BRU	66
1715261751.	ECKE*	67
3321131681.	PFAD0	68
1736133037.	ESAMT	69

LABEL		
004	16	A'
051	17	B'
068	18	C'
076	11	A

Wheatstone-Brücke

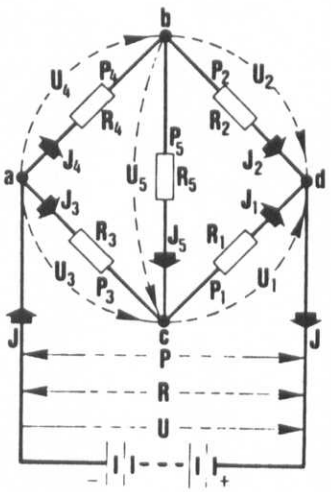


Bild 1

a/d: "Gesamt"
c/d: "PFAD 1"
b/d: "PFAD 2"
a/c: "PFAD 3"
a/b: "PFAD 4"
b/c: "PFAD 5"

(bitte umblättern)

Wheatstone-Brücke

Anmerkung:Steht kein Drucker zur Verfügung,so erfolgt der Abruf jeweils mit "R/S",R,U,J,P usw.

```
--------------------
   EINGABE
--------------------
  R1=
        909.
  R2=
        261.
  R3=
        464.
  R4=
        133.
  R5=
       1870.
   U=
         12.
--------------------
*WHEATSTONE-BRUECKE*
--------------------
    GESAMT
--------------------
 R=
    306.1471 00 OHM
 U=
     12.0000 00 VOLT
 J=
     39.1968-03 AMP.
 P=
    470.3621-03 WATT
--------------------
     PFAD1
--------------------
 R=
    909.0000 00 OHM
 U=
      7.9453 00 VOLT
 J=
      8.7407-03 AMP.
 P=
     69.4470-03 WATT
--------------------
     PFAD2
--------------------
 R=
    261.0000 00 OHM
 U=
      7.9491 00 VOLT
 J=
     30.4562-03 AMP.
 P=
    242.0979-03 WATT
--------------------
     PFAD3
--------------------
 R=
    464.0000 00 OHM
 U=
      4.0547 00 VOLT
 J=
      8.7386-03 AMP.
 P=
     35.4328-03 WATT
--------------------
     PFAD4
--------------------
 R=
    133.0000 00 OHM
 U=
      4.0509 00 VOLT
 J=
     30.4582-03 AMP.
 P=
    123.3843-03 WATT
--------------------
     PFAD5
--------------------
 R=
      1.8700 03 OHM
 U=
      3.7905-03 VOLT
 J=
      2.0270-06 AMP.
 P=
      7.6835-09 WATT
--------------------
```

"A"

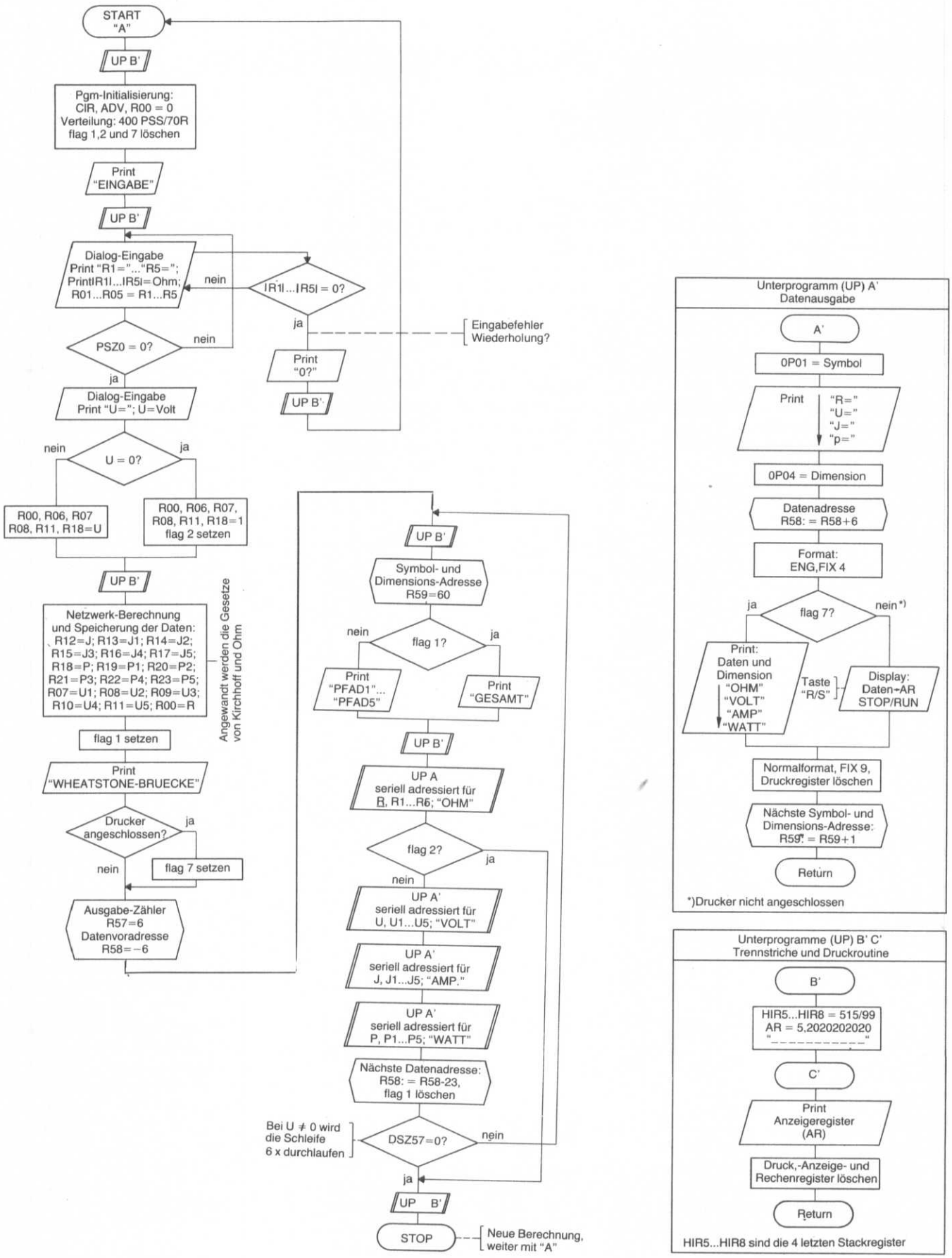
```
306.14714 00    00
      909.  00  01
      261.  00  02
      464.  00  03
      133.  00  04
     1.87   03  05
       12.  00  06
7.9452694   00  07
  7.94906   00  08
4.0547306   00  09
  4.05094   00  10
3.7905417-03    11
39.196839-03    12
8.7406705-03    13
30.456169-03    14
8.7386434-03    15
30.458196-03    16
2.0270276-06    17
470.36207-03    18
69.446982-03    19
242.09791-03    20
35.432845-03    21
123.38432-03    22
7.6835327-09    23
```

```
--------------------
   EINGABE
--------------------
  R1=
       3750.
  R2=
       4300.
  R3=
       7200.
  R4=
       8600.
  R5=
       1000.
   U=
         12.
--------------------
*WHEATSTONE-BRUECKE*
--------------------
    GESAMT
--------------------
 R=
      5.9222 03 OHM
 U=
     12.0000 00 VOLT
 J=
      2.0263-03 AMP.
 P=
     24.3154-03 WATT
--------------------
     PFAD1
--------------------
 R=
      3.7500 03 OHM
 U=
      4.0669 00 VOLT
 J=
      1.0845-03 AMP.
 P=
      4.4106-03 WATT
--------------------
     PFAD2
--------------------
 R=
      4.3000 03 OHM
 U=
      4.0496 00 VOLT
 J=
    941.7699-06 AMP.
 P=
      3.8138-03 WATT
--------------------
     PFAD3
--------------------
 R=
      7.2000 03 OHM
 U=
      7.9331 00 VOLT
 J=
      1.1018-03 AMP.
 P=
      8.7408-03 WATT
--------------------
     PFAD4
--------------------
 R=
      8.6000 03 OHM
 U=
      7.9504 00 VOLT
 J=
    924.4639-06 AMP.
 P=
      7.3498-03 WATT
--------------------
     PFAD5
--------------------
 R=
      1.0000 03 OHM
 U=
    -17.3060-03 VOLT
 J=
    -17.3060-06 AMP.
 P=
    299.4986-09 WATT
--------------------
```

```
--------------------
   EINGABE
--------------------
  R1=
       2260.
  R2=
       4020.
  R3=
       5360.
  R4=
       2150.
  R5=
         0.    ?
--------------------

--------------------
   EINGABE
--------------------
  R1=
       2260.
  R2=
       4020.
  R3=
       5360.
  R4=
       2150.
  R5=
       2870.
   U=
         0.
--------------------
*WHEATSTONE-BRUECKE*
--------------------
    GESAMT
--------------------
 R=
      3.1766 03 OHM
--------------------
```

Das Ausgabeformat(ENG/FIX 4) kann bei Bedarf in PSS 030/ 031/032 geändert werden.

Flußdiagramm "WHEATSTONE-BRUECKE"



Karl Ewe

Sternzeit — Sternposition mit dem TI-59

Beschreibung

Dieses Programm errechnet für einen Beobachtungsort mit der geographischen Breite φ und der geographischen Länge λ zu einem in MGZ (Mittlere Greenwich-Zeit) angegebenen Zeitpunkt die Position eines Sternes mit den Sternkoordinaten Rektazension R und Deklination δ,

den Höhenwinkel H über dem Horizont und

den Richtungswinkel zum Horizontkreis, Azimut Az,

vollkreisig von Nord über Ost usw. gezählt.

Anders ausgedrückt; es gibt Antwort auf die Frage:

Wo „steht" Stern A zum Zeitpunkt B vom Orte C aus beobachtet?

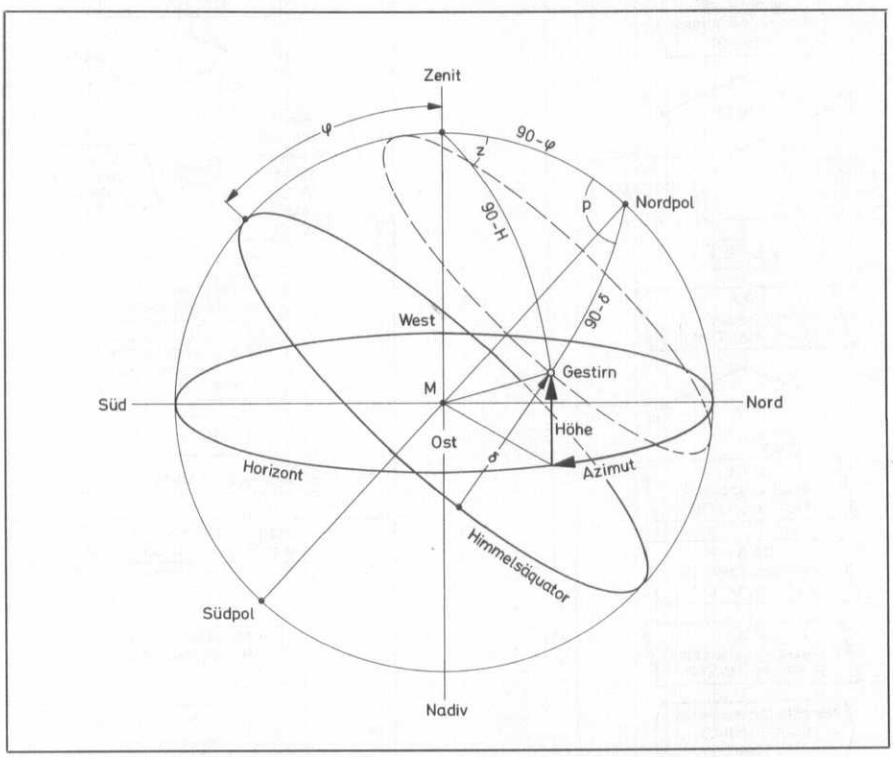
Die verlängerte Erdachse stellt die Himmelsachse dar mit Nord- und Südpol an der Himmelskugel. Der größte Kreis senkrecht zur Achse ist der Himmelsäquator. Die Mitte der waagerechten Ebene des Horizontalkreises liegt in M. Die Gestirne bewegen sich in Parallelkreisen zum Himmelsäquator. Ein Kreis mit Gestirn ist gestrichelt eingezeichnet. Das Gestirn ist vom Beobachter im Mittelpunkt der Himmelskugel über dem Horizont mit dem Höhenwinkel H und dem Horizontalwinkel Azimut, von der Nordrichtung aus gerechnet, zu sehen.

Die Punkte Gestirn — Zenit — Nordpol — Gestirn bilden auf der Himmelskugel ein sphärisches Dreieck. In diesem wird gesucht für H der Bogen Zenit — Gestirn = 90° — H. H ist Komplementwinkel zu Zenit — Gestirn. Dann wird gesucht der Richtungswinkel am Horizont. Dieser ist gleich dem Zenitwinkel Z zwischen den Seiten Zenit — Nordpol und Zenit — Gestirn. Die gesuchten Stücke 90° — H und Z werden mit Hilfe dreier anderer gegebenen Stücke errechnet. Gegeben sind die Seiten Nordpol — Zenit = 90° — φ, und die Seite Nordpol — Gestirn = 90° — δ · φ ist die geographische Breite des Beobachterortes, und δ ist die „himmlische Breite" Deklination des Gestirns. Die beiden vorgenannten Seiten schließen den Polwinkel P ein. Dieser ist indirekt durch den Längengrad λ des Beobachtungsortes und dem Zeitpunkt der Beobachtung gegeben. In diesem Zeitpunkt „steht" das Gestirn senkrecht über dem Längengrad auf der Erde, dem an der Himmelskugel der Bogen Nordpol — Gestirn entspricht. Längengradunterschiede stellen gleichzeitig Zeitunterschiede dar. 360 Winkelgrade = 1 Umdrehung = 1 Tag = 24 Stunden.

Die Zeitdifferenz von einem bestimmten Zeitpunkt bis zu einem anderen Zeitpunkt bedeutet gleicherweise eine Winkeldifferenz von der ersten Winkelstellung bis zur zweiten Winkelstellung. Da gemessen an den Fixsternen die Erde in einem Jahr eine Umdrehung gleich ein Sterntag mehr macht als Sonnentage gezählt werden, geht die Sternzeituhr etwas schneller.

Der Faktor beträgt 366,2422 : 365,2422 = 1,002737909.

Bezugsmarken für Winkel und Zeitmessung sind der Längengrad Null von Greenwich, Großkreis durch Nordpol und Greenwich und an der Himmelskugel der Großkreis durch Nordpol und Frühlingspunkt, ein Schnittpunkt der Jahresbahn der Sonne an der Himmelskugel mit dem Himmelsäquator. Der Winkelunterschied beider Großkreise in gleicher Drehrichtung gemessen in Zeiteinheiten stellt die Sternzeit dar. Sternzeitwinkel, Winkeldifferenz des Gestirns zum Frühlingspunkt, gemessen in Zeit Rektazension R, und der Längengrad des Beobachtungsortes ergeben zusammen den Polwinkel P. Und hiermit sind drei Stücke zur Be-



Zur Erläuterung diene obenstehende Zeichnung. Im Mittelpunkt M der Himmelskugel befindet sich die Erde mit Beobachter, im Verhältnis zur Himmelskugel ein Punkt. Senkrecht über dem Beobachter steht der Zenitpunkt.

rechnung des sphärischen Dreiecks Zenit – Nordpol – Gestirn gegeben.
Die Formel für H lautet

$$H = \arcsin(\sin\varphi \cdot \sin\delta + \cos\varphi \cdot \cos\delta \cdot \cos P)$$

Die Formel für Z lautet

$$Z = \arccos\left(\frac{\sin\delta - \sin\varphi \times \sin H}{\cos\varphi \times \cos H}\right)$$

Auf der Osthälfte der Horizontfläche ist der Horizontalwinkel Az gleich Z, auf der Westhälfte der Horizontfläche ist Az = 360° – Z. Nun zum eigentlichen Programm.
Im Vorprogramm LBL °A' werden der Starttag und dessen Sternzeit, der Faktor 1,0027.., diverse Sternörter R und δ sowie 360° abgespeichert.
Im Nebenprogramm LBL °C' und LBL °D' werden Breite φ und Länge λ des Beobachtungsortes eingegeben und für die Verwendung im Hauptprogramm werden Hilfswerte errechnet. Im Nebenprogramm LBL C und LBL D werden die Sternörter – Sternkoordinaten Rektazension R und Deklination δ eingegeben, und für die Verwendung im Hauptprogramm werden Hilfswerte errechnet.
Das Hauptprogramm beginnt auf LBL A. Hier wird mit Hilfe des Kalenderprogrammes des TI-59, eines Starttages und dessen Sternzeitwinkels und mit dem Faktor 1,0027.. die Sternzeit für den vorgegebenen Tag für 0 Uhr MGZ errechnet und in Stunden und gerundeten Minuten angezeigt. (Wie im Kosmos-Bändchen „Das Himmelsjahr" die Sternzeit angegeben wird.) Hierauf wird das Hauptprogramm auf LBL B fortgesetzt mit der Umrechnung einer Uhrzeit des vorgegebenen Tages von MGZ in Sternzeit. Diese wird angezeigt. Ein Unterprogramm SBR 147 dient zur Kürzung der Anzeige auf gerundete Minuten. Nun wird auf LBL E das Hauptprogramm fortgesetzt. Der zur Höhenformel erforderliche Polwinkel P wird bestimmt und der Höhenwinkel H errechnet und in Deg angezeigt. Dann wird nach Taste R/S der Horizontalwinkel Azimut Az errechnet und in Deg angezeigt. Ein Unterprogramm SBR 172 rundet auf 1/100 Grad.

Gebrauchsanweisung

Nach Programmierung		Taste	Anzeige
Vorbereitung		°A'	360
Eingaben			
Beobach-	φ Grad, min (s)	°C'	φ
tungsort	Nord + Süd –		
	λ West + Ost –	°D'	λ
Datum	MMTT.JJJJ	A	* t für 0 h MGZ
Uhrzeit	MGZ h, min (s)	B	* t für Uhrzeit MGZ
Sternort	Rektazension R h, min s	C	R
	Deklination δ Grad, min s	D	δ
Rechnen	Höhe H	E	H Deg
	Azimut Az	R/S	Az Deg

Im Programm sind die Sternörter von elf hellen Sternen gespeichert.

Sternörter im Programm

Stern		R in M	δ in M	Rektazension R			Deklination δ		
Aldebaran	α Stier	31	32	14 h	34 min	50 s	16°	28'	18"
Arcturus	α Bootes	33	34	14 h	14 min	48 s	19°	16'	48"
Atair	α Adler	35	36	19 h	49 min	52 s	8°	49'	6"
Beteigeuze	α Orion	37	38	5 h	54 min	9 s	7°	24'	12"
Capella	α Fuhrmann	39	40	5 h	15 min	18 s	45°	58'	42"
Deneb	α Schwan	41	42	20 h	40 min	48 s	45°	12'	48"
Gemma	α Ndl. Krone	43	44	15 h	33 min	53 s	26°	46'	42"
Regulus	α Löwe	45	46	10 h	7 min	22 s	12°	3'	30"
Sirius	α Gr. Hund	47	48	6 h	44 min	19 s	–16°	41'	30"
Spica	α Jungfrau	49	50	13 h	24 min	12 s	–11°	3'	48"
Wega	α Leier	51	52	18 h	36 min	18 s	38°	45'	0"

(bitte umblättern)

Sternzeit – Sternposition mit dem TI-59

Beispiel 1: Wo „steht" Stern Regulus am 15. 3. 1981 von München, Breite φ 48°9' Länge – 11°36', aus beobachtet um 20 h 50 min MGZ?

			Taste	Anzeige			
Nach Programmierung Vorbereitung			A'	360			
Eingaben							
Ort	München						
Breite φ		48,09	C'	48,09			
Länge λ		−11,36	D'	−11,36			
Datum	15. 3. 1981	15.1981	A	11,3	*t h min	für 0 h MGZ	
Uhrzeit		20.50	B	8,23	*t	für Uhrzeit	
Stern	R RCL 45 Anz.	10.0722	C	10.0722			
Regulus	δ RCL 46	12.033	D	12.033			
Rechnen	Höhe H		E	51,97	Antwort	Höhe	51,97° Deg
	Azimut Az		R/S	156,79		Azimut	156,79°
und wo um 23 h 12 min MGZ?							
Eingabe		23,12	B	10.46			
Rechnen	Höhe H		E	49,8	Antwort	Höhe	49,8° Deg
	Azimut Az		R/S	213,26		Azimut	213,26°

Beispiel 2: Wo „steht" Stern Arcturus am 1. 8. 1981 von München aus beobachtet um 20 h 30 min MGZ?

Eingaben							
Datum	1. 8. 1981	801.1981	A	20,38	*t	0 h MGZ	
Uhrzeit		20.30	B	17,11	*t		
Stern	R RCL 33 Anz	14.1448	C	14,1448			
Arcturus	δ RCL 34	19.1648	D	19.1648			
Rechnen	Höhe H		E	36,89	Antwort	Höhe	36,89° Deg
	Azimut Az		R/S	257,33		Azimut	257,33°
und wo um 21 h 33 min 45 s MGZ?							
Eingabe		21.3345	B	18.15			
Rechnen	Höhe H		E	26,31	Antwort	Höhe	26,31° Deg
	Azimut Az		R/S	270.		Azimut	270°

Speicherplan

1–5	belegt mit Kalender Pgm 20	14	sin φ	21	Deklination δ dieses Gestirns: in Deg
6	1. 1. 1980	15	cos φ	22	sin δ
7	1,002737909			23	cos δ
8	0,277251157	16	Länge λ des Beobachtungsortes Grad min s	24	λ^*_0
9	360	17	Länge λ des Beobachtungsortes in Deg	25	λ^*
10	Sternzeit für Oh GMT des Datums als Dezibruch für λ 0°	18	Rectazension R h min s eines Gestirnes	26	$\Delta\lambda$ = (λ Beobachtungsort $-\lambda^*$) = $\sphericalangle$ P
11	Sternzeit für eine andere GMT dieses Tages als Dezibruch für λ 0°	19	Sternwinkel dieses Gestirns als Dezibruch (1 – R) R hier als Dezibruch	27	Höhenwinkel H des Gestirns Deg
12	Breite φ des Beobachtungsortes Grad min s	20	Deklination δ dieses Gestirns Grad min s	28	Zenitwinkel Z (Azimut halbkreisig) Deg
13	Breite φ des Beobachtungsortes in Deg			29	Azimut Az vollkreisig Deg

R	δ	Stern	im Sternbild	R			δ		
31	32	Aldebaran	α Stier	4 h	34 min	50 s	16°	28'	18″
33	34	Arcturus	α Bootes	14 h	14 min	48 s	19°	16'	48″
35	36	Atair	α Adler	19 h	49 min	52 s	8°	49'	06″
37	38	Beteigeuze	α Orion	5 h	54 min	9 s	7°	24'	12″
39	40	Capella	α Fuhrmann	5 h	15 min	18 s	45°	58'	42″
41	42	Deneb	α Schwan	20 h	40 min	48 s	45°	12'	48″
43	44	Gemma	α Ndl. Krone	15 h	33 min	53 s	26°	46'	42″
45	46	Regulus	α Löwe	10 h	7 min	22 s	12°	03'	30″
47	48	Sirius	α Gr. Hund	6 h	44 min	19 s	−16°	41'	30″
49	50	Spica	α Jungfrau	13 h	24 min	12 s	−11°	3'	48″
51	52	Wega	α Leier	18 h	36 min	18 s	38°	45'	0″

(bitte umblättern)

Das Text-Genie. Der GENIETEXT.

GENIETEXT – die neue professionelle Software. Mit GENIETEXT machen Sie aus Ihrem Video-Genie ein Text-Genie. GENIETEXT – ein Textsystem, das speziell für den deutschen Markt konzipiert ist. GENIETEXT – das Textsystem von Profis, für Profis. Mehr als ein Spielzeug. GENIETEXT – auf Diskette gespeichert. Nur in das Laufwerk eingeben – und schon geht's los. Geräte-Konfiguration: Video-Genie mit Expansions-Interface (48 K), 2 Diskdrives, Drucker. Hardware-Baustein: Kleinschriftmodul mit deutschen Umlauten und „ß". Einfache Montage – Kompl. DM 145,– incl. MWSt. GENIETEXT – original Software auf Diskette, Übungstext und deutschem Handbuch, DM 395,– incl. MWST.

Trommeschläger
Computerstudio GmbH
Flugplatz Bonn-Hangelar
Postfach 2105, 5205 St. Augustin 2
Telefon (0 22 41) 2 00 61-62
Alleinvertrieb der EACA-Produkte

Fachbücher für Technik und Wirtschaft

Katalog liefert
VOGEL-VERLAG
Postfach 67 40
8700 Würzburg 1

Technik, Wirtschaft, Universal

Zahlreiche Titel der Fächer Mathematik, Physik, Chemie, Kunststoffe/-Verarbeitung, Elektrotechnik, Elektronik, Daten verarbeitung, Computertechnik, Maschinenbau, Kfz- und Landmaschinentechnik, Fertigungstechnik, Betriebsorganisation, Wirtschaft

SEMINARE über die Anwendung der
HEWLETT-PACKARD-RECHNER HP 41C + HP 85
ganzjährig im gesamten Bundesgebiet.

Weiterhin bieten wir interessantes **Zubehör:**
z.B. handlicher, stabiler Koffer für das vollständige HP-41C System.
Professionelle Software u. a. aus den Bereichen: Effektivzinsberechnung – Lohn-/Einkommensteuer – Festigkeitsberechnung – Statistik – Karteiverwaltung/Textverarbeitung etc.
Fordern Sie unsere kostenlosen Informationsbroschüren an!
Petunienweg 1 · 6382 Friedrichsdorf · Telefon (06172) 79551

CORVALLIS TEAM
SEMINAR & SOFTWARE SERVICE GMBH
CT

Sternzeit — Sternposition mit dem TI-59

```
000  76  LBL     040  43  RCL     080  43  RCL     120  65  ×       160  00  00      200  42  STO     240  17  17      280  04  4       320  04  4       360  04  4       400  02  2
001  11  A       041  15  15      081  15  15      121  43  RCL     161  52  EE      201  20  20      241  43  RCL     281  42  STO     321  09  9       361  05  5       401  42  STO
002  36  PGM     042  65  ×       082  65  ×       122  27  27      162  22  INV     202  88  DMS     242  16  16      282  08  08      322  05  5       362  08  8       402  44  44
003  20  20      043  43  RCL     083  43  RCL     123  36  COS     163  52  EE      203  42  STO     243  91  R/S     283  04  4       323  02  2       363  04  4       403  01  1
004  12  B       044  20  20      084  20  20      124  95  =       164  55  ÷       204  27  27      244  76  LBL     284  93  .       324  42  STO     364  02  2       404  00  0
005  36  PGM     045  65  ×       085  65  ×       125  22  INV     165  01  1       205  38  SIN     245  11  A       285  03  3       325  35  35      365  42  STO     405  93  .
006  20  20      046  53  (       086  53  (       126  35  COS     166  00  0       206  42  STO     246  36  PGM     286  04  4       326  08  8       366  40  40      406  00  0
007  13  C       047  43  RCL     087  43  RCL     127  42  STO     167  00  0       207  22  22      247  20  20      287  05  5       327  93  .       367  02  2       407  07  7
008  65  ×       048  17  17      088  17  17      128  28  28      168  95  =       208  43  RCL     248  01  1       288  42  STO     328  04  4       368  00  0       408  02  2
009  43  RCL     049  75  -       089  75  -       129  43  RCL     169  22  INV     209  21  21      249  00  0       289  31  3       329  09  9       369  93  .       409  02  2
010  07  07      050  43  RCL     090  43  RCL     130  26  26      170  58  FIX     210  39  COS     250  01  1       290  01  1       330  00  0       370  04  4       410  42  STO
011  85  +       051  25  25      091  25  25      131  38  SIN     171  92  RTN     211  42  STO     251  93  .       291  06  6       331  06  6       371  00  0       411  45  45
012  43  RCL     052  54  )       092  54  )       132  77  GE      172  58  FIX     212  23  23      252  01  1       292  93  .       332  42  STO     372  04  4       412  01  1
013  08  08      053  42  STO     093  42  STO     133  01  0.      173  02  02      213  43  RCL     253  09  9       293  02  2       333  36  36      373  08  8       413  02  2
014  95  =       054  65  ×       094  26  26      134  38  38      174  52  EE      214  20  20      254  08  8       294  08  8       334  05  5       374  42  STO     414  93  .
015  22  INV     055  43  RCL     095  35  COS     135  43  RCL     175  22  INV     215  91  R/S     255  00  0       295  01  1       335  93  .       375  41  41      415  00  0
016  59  INT     056  09  09      096  95  =       136  09  09      176  52  EE      216  76  LBL     256  11  A       296  08  8       336  05  5       376  04  4       416  03  3
017  42  STO     057  95  =       097  22  INV     137  75  -       177  22  INV     217  18  C'      257  01  1       297  42  STO     337  04  4       377  05  5       417  03  3
018  10  10      058  42  STO     098  38  SIN     138  43  RCL     178  58  FIX     218  42  STO     258  93  .       298  32  32      338  00  0       378  93  .       418  42  STO
019  71  SBR     059  24  24      099  42  STO     139  28  28      179  92  RTN     219  12  12      259  00  0       299  01  1       339  09  9       379  01  1       419  46  46
020  01  01      060  38  SIN     100  27  27      180  76  LBL     220  88  DMS     260  00  0       300  04  4       340  42  STO     380  02  2       420  06  6
021  47  47      061  29  CP      101  71  SBR     181  13  C       221  42  STO     261  02  2       301  93  .       341  37  37      381  04  4       421  93  .
022  91  R/S     062  77  GE      102  01  01      182  42  STO     222  13  13      262  07  7       302  01  1       342  07  7       382  08  8       422  04  4
023  76  LBL     063  00  00      103  72  ST      183  18  18      223  38  SIN     263  03  3       303  36  36      343  93  .       383  42  STO     423  04  4
024  12  B       064  69  69      104  91  R/S     184  88  DMS     224  42  STO     264  07  7       304  05  5       344  02  2       384  42  42      424  01  1
025  88  DMS     065  43  RCL     105  53  (       185  55  ÷       225  14  14      265  09  9       305  93  .       345  04  4       385  01  1       425  09  9
026  55  ÷       066  09  09      106  43  RCL     186  02  2       226  43  RCL     266  00  0       306  05  5       346  01  1       386  05  5       426  42  STO
027  02  2       067  94  +/-     107  22  22      187  04  4       227  13  13      267  09  9       307  04  4       347  02  2       387  00  0       427  47  47
028  04  4       068  85  +       108  75  -       188  95  =       228  36  COS     268  42  STO     308  00  0       348  42  STO     388  93  .       428  01  1
029  65  ×       069  43  RCL     109  43  RCL     189  94  +/-     229  42  STO     269  07  07      309  09  9       349  38  38      389  03  3       429  06  6
030  43  RCL     070  24  24      110  14  14      190  85  +       230  15  15      270  00  0       310  93  .       350  05  5       430  93  .
031  07  07      071  95  =       111  65  ×       191  01  1       231  43  RCL     271  93  .       311  01  1       351  93  .       431  04  4
032  85  +       072  42  STO     112  43  RCL     192  95  =       232  12  12      272  02  2       312  06  6       352  01  1       432  01  1
033  43  RCL     073  25  25      113  27  27      193  42  STO     233  91  R/S     273  07  7       313  04  4       353  05  5       433  00  0
034  10  10      074  43  RCL     114  38  SIN     194  19  19      234  76  LBL     274  07  7       314  08  8       354  01  1       434  94  +/-
035  95  =       075  14  14      115  54  )       195  43  RCL     235  15  E       275  02  2       315  42  STO     355  08  8       435  42  STO
036  22  INV     076  65  ×       116  55  ÷       196  18  18      236  42  STO     276  05  5       316  34  34      356  42  STO     436  48  48
037  59  INT     077  43  RCL     117  53  (       197  91  R/S     237  16  16      277  01  1       317  01  1       357  39  39      437  01  1
038  42  STO     078  22  22      118  43  RCL     198  76  LBL     238  88  DMS     278  01  1       318  09  9       358  04  4       438  03  3
039  11  11      079  85  +       119  15  15      199  14  D       239  42  STO     279  05  5       319  93  .       359  05  5       439  93  .
040  71  SBR     080  43  RCL     120  65  ×       200  42  STO     280  07  7       320  04  4       360  93  .       440  02  2
041  01  01      081  15  15      121  43  RCL     201  20  20      281  42  STO     321  09  9       361  05  5       441  04  4
042  47  47      082  65  ×       122  27  27      202  88  DMS     282  08  08      322  05  5       362  08  8       442  01  1
043  91  R/S     083  43  RCL     123  36  COS     203  42  STO     283  04  4       323  02  2       363  04  4       443  02  2
044  76  LBL     084  20  20      124  95  =       204  21  21      284  93  .       324  42  STO     364  02  2       444  42  STO
045  15  E       085  65  ×       125  22  INV     205  38  SIN     285  03  3       325  49  49      365  42  STO     445  49  49
046  43  RCL     086  53  (       126  35  COS     206  42  STO     286  04  4       326  40  40      366  40  40      446  01  1
047  11  11      087  43  RCL     127  42  STO     207  22  22      287  05  5       327  02  2       367  02  2       447  01  1
048  85  +       088  17  17      128  28  28      208  43  RCL     288  42  STO     328  00  0       368  00  0       448  93  .
049  43  RCL     089  75  -       129  43  RCL     209  21  21      289  31  3       329  93  .       369  93  .       449  00  0
050  19  19      090  43  RCL     130  26  26      210  39  COS     290  01  1       330  04  4       370  04  4       450  03  3
051  95  =       091  25  25      131  38  SIN     211  42  STO     291  06  6       331  00  0       371  00  0       451  04  4
052  22  INV     092  54  )       132  77  GE      212  23  23      292  93  .       332  04  4       372  04  4       452  08  8
053  59  INT     093  42  STO     133  01  0.      213  43  RCL     293  02  2       333  08  8       373  08  8       453  94  +/-
054  65  ×       094  26  26      134  38  38      214  20  20      294  08  8       334  42  STO     374  42  STO     454  42  STO
055  43  RCL     095  35  COS     135  43  RCL     215  91  R/S     295  01  1       335  41  41      375  41  41      455  50  50
056  09  09      096  95  =       136  09  09      216  76  LBL     296  08  8       336  04  4       376  04  4       456  01  1
057  95  =       097  22  INV     137  75  -       217  18  C'      297  42  STO     337  05  5       377  05  5       457  08  8
058  42  STO     098  38  SIN     138  43  RCL     218  42  STO     298  32  32      338  93  .       378  93  .       458  93  .
059  24  24      099  42  STO     139  28  28      219  12  12      299  01  1       339  01  1       379  01  1       459  03  3
060  38  SIN     100  95  =       140  95  =       220  88  DMS     300  04  4       340  02  2       380  02  2       460  06  6
061  29  CP      101  42  STO     141  42  STO     221  43  STO     301  93  .       341  04  4       381  04  4       461  01  1
062  77  GE      102  29  29      142  29  29      222  13  13      302  01  1       342  08  8       382  08  8       462  08  8
063  00  00      103  71  SBR     143  71  SBR     223  38  SIN     303  04  4       343  42  STO     383  42  STO     463  42  STO
064  69  69      104  01  01      144  01  01      224  42  STO     304  04  4       344  42  42      384  42  42      464  51  51
065  43  RCL     105  72  ST      145  72  ST      225  14  14      305  08  8       345  01  1       385  01  1       465  03  3
066  09  09      106  91  R/S     146  91  R/S     226  43  RCL     306  42  STO     346  05  5       386  05  5       466  08  8
067  94  +/-     107  65  ×       147  65  ×       227  13  13      307  30  30      347  93  .       387  93  .       467  90  ;
068  85  +       108  02  2       148  02  2       228  36  COS     308  01  1       348  03  3       388  03  3       468  04  4
069  43  RCL     109  04  4       149  04  4       229  42  STO     309  09  9       349  03  3       389  03  3       469  05  5
070  24  24      110  95  =       150  95  =       230  15  15      310  93  .       350  05  5       390  05  5       470  42  STO
071  95  =       111  22  INV     151  22  INV     231  43  RCL     311  01  1       351  03  3       391  03  3       471  52  52
072  42  STO     112  88  DMS     152  88  DMS     232  12  12      312  06  6       352  42  STO     392  42  STO     472  03  3
073  25  25      113  65  ×       153  65  ×       233  91  R/S     313  04  4       353  43  43      393  43  43      473  06  6
074  43  RCL     114  01  1       154  01  1       234  76  LBL     314  08  8       354  02  2       394  02  2       474  00  0
075  14  14      115  00  0       155  00  0       235  19  D'      315  42  STO     355  06  6       395  06  6       475  42  STO
076  65  ×       116  00  0       156  00  0       236  42  STO     316  34  34      356  93  .       396  93  .       476  09  09
077  43  RCL     117  95  =       157  95  =       237  16  16      317  01  1       357  04  4       397  04  4       477  91  R/S
078  22  22      118  88  DMS     158  88  DMS     238  88  DMS     318  09  9       358  06  6       398  06  6       478  00  0
079  85  +       119  58  FIX     159  58  FIX     239  42  STO     319  93  .       359  04  4       399  04  4       479  00  0
```

Enno Apley

TI-59 berechnet Elektronenkonfiguration der Elemente

Beschreibung

Nach der einzig notwendigen Eingabe der Ordnungszahl des zu untersuchenden Elementes über Label a druckt das Programm optisch effektvoll auf der rechten Streifenseite die Bezeichnung der Energieniveaus, 1s, 2s, 2p usw., und außerdem in der Mitte die Anzahl der dort besetzten Elektronen. Dieses geschieht, indem der Rechner die Reihenfolge, in der die verschiedenen Energiezustände aufgefüllt werden, simuliert. Diese Tatsache macht das Programm im Vergleich zu Tabellenwerken interessant, da aus jenen die Reihenfolge und das Prinzip der Elektronenbesetzung nicht ersichtlich ist. Das Programm läßt sich auf fast alle Elemente anwenden, allerdings weisen einige wenige Elemente (Lanthaniden) Anomalien auf und genügen aus energetischen Gründen dem dem Programm zugrundeliegenden Prinzip nicht.

Legt man das Schema der Elektronenkonfiguration* auf ein Koordinatensystem, wobei die Hauptquantenzahlen (entspricht K, L, M usw.-Schale) auf der Ordinate, die spdf-Orbitale auf der Abzisse liegen, dann erhält man eine mathematische Folge von Zahlenpaaren, wenn man die Pfeile entlangfährt:

1/1
2/1
2/2
3/1
3/2
usw.

Der eigentlich wesentliche Teil des Programmes erzeugt im Hauptprogramm und im Subprogramm nun diese Folge. Die x-Koordinate kann gleich in die maximale Zahl der Elektronen dieses Orbitals umgerechnet werden, wie dies in Zeile 58—67 geschieht. Die Ordnungszahl ist ja nichts anderes als die Anzahl der Elektronen des Elementes. Diese, in Register 4 abgelegt, wird nun um die max. Elektronenzahl vermindert, wobei das Programm mit seinen Koordinaten immer weiter „wandert", bis R_4 Null ist oder kleiner als die max. Elektr.-Zahl in R_3.

Es wird dann im Unterprogramm gestoppt, beim erneuten Programmstart allerdings das UPGM-Rücksprungstackregister durch „Reset" gelöscht.

Um die Hauptquantenzahl in Register 2 durch OP 04 und OP 06 auf der rechten Druckseite ausgeben zu können, muß sie in den alphanumerischen Kode des TI-59 verwandelt werden.

Dies geschieht in den Programmzeilen 031—045.

Die x-Werte in R_1 finden ihre Entsprechung als Kode in den Speichern 10—15, diese werden über indirekte Adressierung von R_3 richtig zugeordnet.

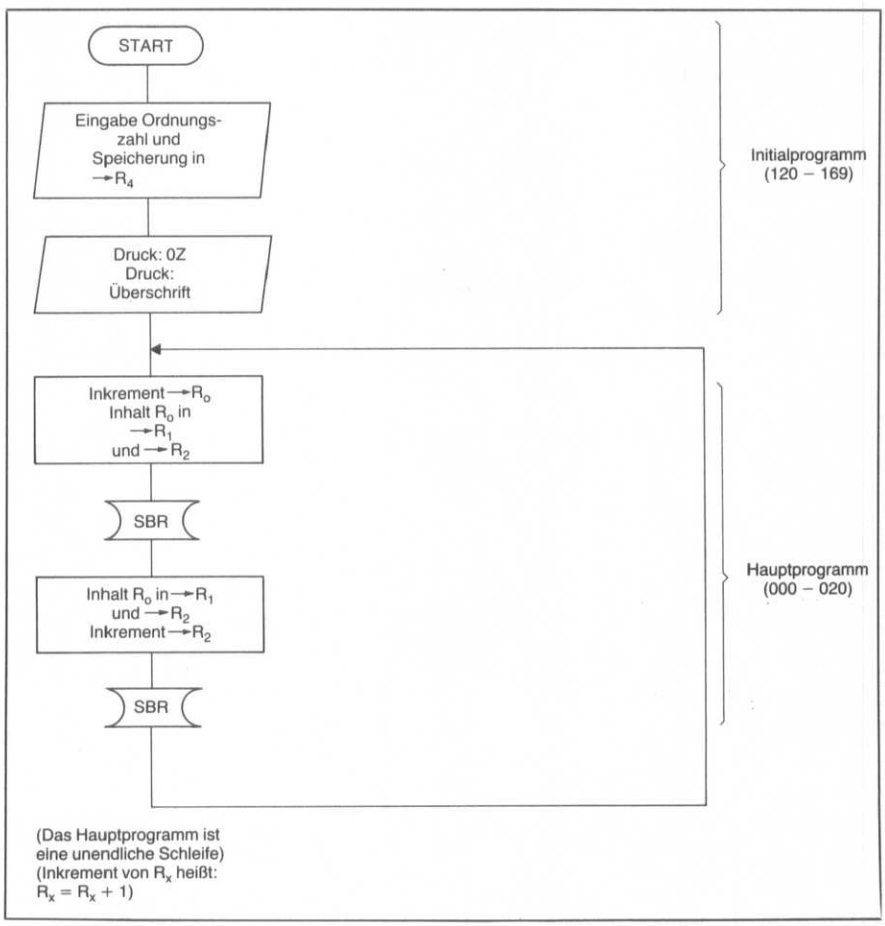
Benutzeranweisung:

1. Programm eingeben und auf Magnetkarten speichern (Block 1).
2. Daten in Speicher 05—15 eingeben und auf Magnetkarte speichern (Block 4).
3. Ordnungszahl eintippen und Label A drücken.

Verbesserungsvorschlag!

Damit das PGM richtig arbeitet, muß das t-Register 0. enthalten.

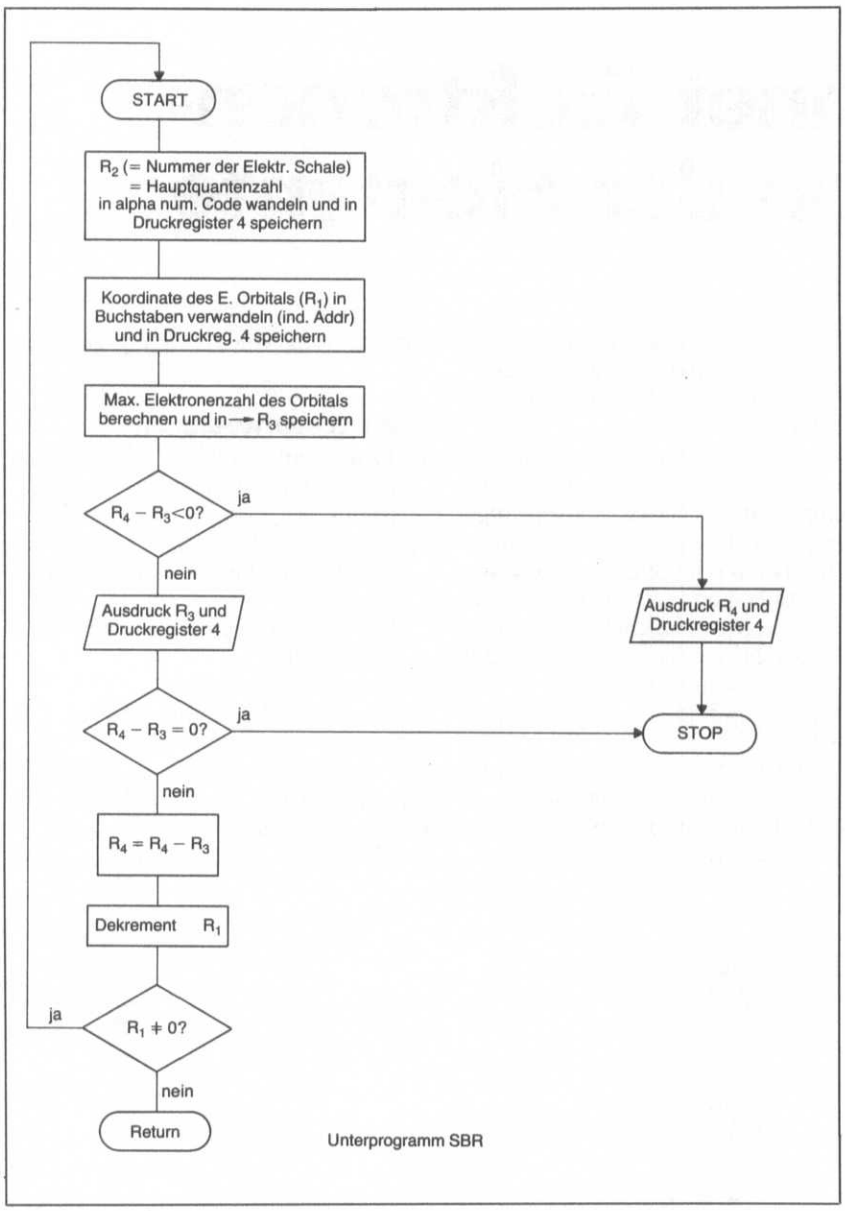
Um dies auf jeden Fall zu gewährleisten, würde ich am Schluß noch den Befehl CP einfügen.



(bitte umblättern)

* Siehe dazu z.B.: H. R. Christen, Grundlagen der allgemeinen und anorganischen Chemie.

TI-59 berechnet Elektronenkonfiguration der Elemente




```
000  69 OP
001  20  20
002  43 RCL
003  00  00
004  42 STO
005  01  01
006  42 STO
007  02  02
008  71 SBR
009  71 SBR
010  43 RCL
011  00  00
012  42 STO
013  01  01
014  42 STO
015  02  02
016  69 OP
017  22  22
018  71 SBR
019  71 SBR
020  81 RST
021  76 LBL
022  71 SBR
023  43 RCL
024  01  01
025  85  +
026  01  1
027  00  0
028  95  =
029  42 STO
030  03  03
031  43 RCL
032  02  02
033  85  +
034  53  (
035  24 CE
036  75  -
037  06  6
038  93  .
039  05  5
040  54  )
041  69 OP
042  10  10
043  85  +
044  02  2
045  95  =
046  65  ×
047  01  1
048  00  0
049  00  0
050  00  0
051  00  0
052  85  +
053  73 RC*
054  03  03
055  95  =
056  69 OP
057  04  04
058  02  2
059  65  ×
060  53  (
061  43 RCL
062  01  01
063  65  ×
064  02  2
065  75  -
066  01  1
067  95  =
068  42 STO
069  03  03
070  94 +/-
071  85  +
072  43 RCL
073  04  04
074  95  =
075  22 INV
076  77  GE
077  00  00
078  99  99
079  43 RCL
080  03  03
081  69 OP
082  06  06
083  94 +/-
084  85  +
085  43 RCL
086  04  04
087  95  =
088  67  EQ
089  01  01
090  03  03
091  42 STO
092  04  04
093  69 OP
094  22  22
095  97 DSZ
096  01  01
097  71 SBR
098  92 RTN
099  43 RCL
100  04  04
101  69 OP
102  06  06
103  43 RCL
104  10  10
105  69 OP
106  01  01
107  69 OP
108  02  02
109  69 OP
110  03  03
111  69 OP
112  04  04
113  69 OP
114  05  05
115  98 ADV
116  98 ADV
117  98 ADV
118  25 CLR
119  91 R/S
120  76 LBL
121  11  A
122  42 STO
123  04  04
124  43 RCL
125  05  05
126  69 OP
127  04  04
128  43 RCL
129  04  04
130  69 OP
131  06  06
132  25 CLR
133  42 STO
134  00  00
135  69 OP
136  00  00
137  69 OP
138  05  05
139  43 RCL
140  10  10
141  69 OP
142  01  01
143  69 OP
144  02  02
145  69 OP
146  03  03
147  69 OP
148  04  04
149  69 OP
150  05  05
151  43 RCL
152  06  06
153  69 OP
154  01  01
155  43 RCL
156  07  07
157  69 OP
158  02  02
159  43 RCL
160  08  08
161  69 OP
162  03  03
163  43 RCL
164  09  09
165  69 OP
166  04  04
167  69 OP
168  05  05
169  81 RST
170  00  0
```

```
        0.      04
    14640.      05
   542013.      06
3146132327.     07
  62006300.     08
 135144062.     09
5151515151.     10
       36.      11
       33.      12
       16.      13
       21.      14
       22.      15
        0.      16
```

```
162  03  3
163  43 RCL
164  09  09
165  69 OP
166  04  04
167  69 OP
168  05  05
169  29 CP
170  81 RST
171  00  0
172  00  0
173  00  0
```

Beispiel: Ausdruck für das Element Schwefel

```
        16.      0Z.

********************
  e-ANZAHL ; / ORB.;
        2.      1 S
        2.      2 S
        6.      2 P
        2.      3 S
        4.      3 P
********************
```

Karl Silber

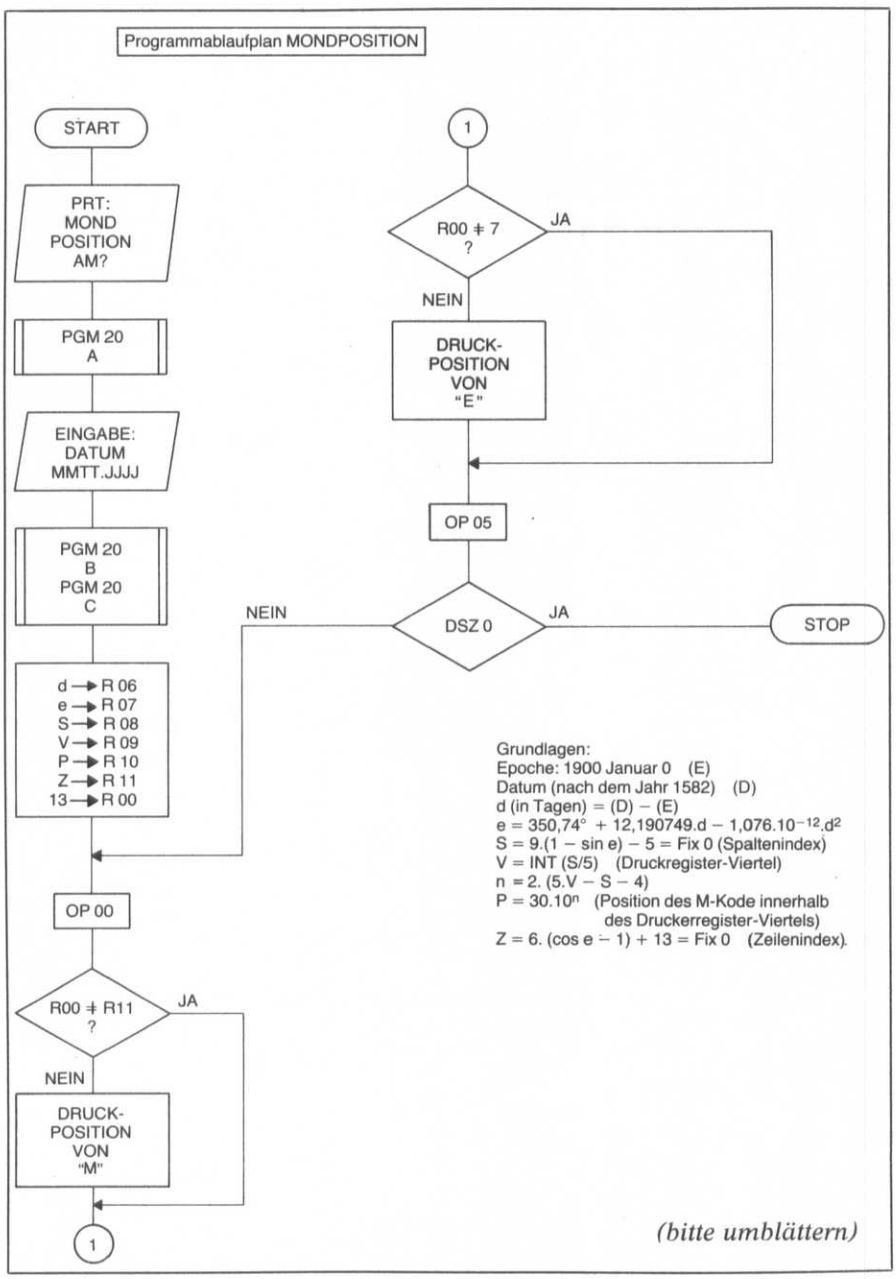
Mondposition mit TI-58/59

Beschreibung

Für die wechselnde Lichtgestalt des Mondes ist sein Umlauf um die Erde und die damit verbundene ständige Änderung seiner Position relativ zu Sonne und Erde verantwortlich. Stets ist nur eine Hälfte des Mondes von der Sonne beleuchtet. Diesen hellen Teil können wir nur dann zur Gänze sehen, wenn Sonne und Mond, von der Erde aus betrachtet, am Himmel gegenüberstehen. Ein kleiner Neigungswinkel zwischen den Ebenen der Mondbahn und der jährlichen Bahn der Erde um die Sonne verhindert es, daß bei dieser Vollmondstellung der Erdschatten jedesmal den Mond verdeckt und eine Mondfinsternis erzeugt. Nach einem halben Umlauf um die Erde gelangt unser Trabant dann in die Neumondphase: Er befindet sich zwischen Sonne und Erde, die uns zugewandte Mondhälfte ist unbeleuchtet. Auf dem Weg von der Vollmond- zur Neumondposition nimmt dabei der Anteil, den wir von der beleuchteten Mondhälfte sehen, ständig ab. Wenn sich der Mond auf dem gegenüberliegenden Stück seiner Bahn bewegt (Richtung Vollmond), nimmt dieser Anteil ständig zu.

Die Bahn des Mondes um die Erde ist eine Ellipse, die dadurch bedingte veränderliche Bahngeschwindigkeit und eine Reihe von anderen Faktoren komplizieren die Erfassung der Mondbewegung. Ihre exakte formelmäßige Darstellung zählt zu den schwierigsten Aufgaben in der Astronomie.

Begnügt man sich mit einer verminderten Genauigkeit, um einen Überblick über die jeweilige Position des Mondes auf seiner Bahn um die Erde und damit auch eine Information über die Mondphase zu erhalten, kann eine gleichmäßige Mondbewegung vorausgesetzt werden. Wird die Sonne als Meßmarke betrachtet, so dauert ein Umlauf des Mondes um die Erde 29,530589 Tage. Diese Zeitspanne zwischen zwei aufeinanderfolgenden Neumondphasen wird als synodischer Monat oder Lunation bezeichnet. Für einen irdischen Beobachter nimmt daher der Winkelabstand zwischen Sonne und Mond, die Elongation, täglich im Mittel um 360° : 29,530589 = 12,190749° zu. Damit sind auch schon die Grundlagen für unser vereinfachtes Mondbahnmodell geschaffen: Die Erde befindet sich im Koordinatenursprung, die Nullrichtung weist zur Sonne, der Mond umkreist mit gleichmäßiger Geschwindigkeit im modellbedingten Abstand die Erde. Da die Bewegung des Mondes auf seiner Bahn um die Erde von Westen nach Osten erfolgt (rechtsläufige Bewegung), für einen Beobachter auf der nördlichen Erdhälfte also entgegen dem Uhrzeigersinn, entspricht sie auch der positiven Zählrichtung im Polarkoordinatensystem. Man kann nun einen Neumondtermin als Ausgangszeitpunkt wählen oder von einer Epoche ausgehen. In der Astronomie wird häufig die Epoche 1900 Januar 0,5 WZ



(bitte umblättern)

Mondposition mit TI-58/59

(=31. Dezember 1899, 12^h WZ) verwendet. Im Programm ist aus Gründen der Vereinfachung 1900 Januar 0 gewählt worden, daher bezieht sich jede Position auf 12^h WZ = 13^h MEZ des betreffenden Datums. Bezeichnet man ein beliebiges Datum (nach 1582) mit D und die Epoche mit E, so ist die Tagesdifferenz d = D − E. Mit dem Phasenwinkel von 350,74° für die Epoche und bei Beachtung einer säkularen Variation erhält man schließlich die Elongation e für ein bestimmtes Datum aus

$$e = 350{,}74° + 12{,}190\,749° \cdot d - 1{,}076° \cdot 10^{-12} \cdot d^2$$

Für die Druckpositionen benötigt man noch

S = 9 · (1 − sin e) + 5 = Fix 0 (Spaltenindex)

Z = 6 · (cos e − 1) + 13 = Fix 0 (Zeilenindex)

V = INT (S/5) (Druckerregister-Viertel)

n = 2 · (5 · V − S + 4)

P = $30 \cdot 10^n$ (Position des M-Kodes innerhalb des Druckerregister-Viertels)

Das Programm wird durch Drücken der Taste A gestartet. Nach der Eingabe eines beliebigen Datums (nach dem Jahr 1582) in der Form MMTT.JJJJ und Drücken der R/S-Taste wird die Stellung des Mondes (M) auf seiner Bahn um die Erde (E) für das eingegebene Datum ausgedruckt. Die Richtung zur Sonne (S↑) wird ebenfalls angezeigt (Richtung des Papiervorschubs). Die Breite des Papierstreifens entspricht dem Durchmesser der Mondbahn. Die jeweilige Phase des Mondes läßt sich nach den vorangegangenen Ausführungen unschwer bestimmen (Genauigkeit ±1^d). Die Datenregister 0 und 6 bis 11 werden durch das Programm selbst belegt, die Register 1 bis 5 über das benutzte Modulprogramm ML-20.

```
000  76 LBL    059  99 PRT    118  00  00    177  00  00    188  00  00    199  10  10    210  02  02
001  11  A     060  98 ADV    119  52 EE     178  03  3     189  43 RCL    200  84 □P*   211  69 □P
002  69 □P     061  22 INV    120  22 INV    179  06  6     190  00  00    201  09  09    212  05  05
003  00  00    062  58 FIX    121  52 EE     180  06  6     191  32 X↕T    202  07  7     213  97 DSZ
004  03  3     063  36 PGM    122  42 STO    181  00  0     192  43 RCL    203  22 INV    214  00  00
005  00  0     064  20  20    123  08  08    182  69 □P     193  11  11    204  67  EQ    215  01  01
006  69 □P     065  12  B     124  55  ÷     183  02  02    194  22 INV    205  02  02    216  87  87
007  01  01    066  36 PGM    125  05  5     184  69 □P     195  67  EQ    206  11  11    217  25 CLR
008  03  3     067  20  20    126  95  =     185  05  05    196  02  02    207  01  1     218  98 ADV
009  02  2     068  13  C     127  59 INT    186  98 ADV    197  02  02    208  07  7     219  98 ADV
010  03  3     069  42 STO    128  42 STO    187  69 □P     198  43 RCL    209  69 □P     220  91 R/S
011  01  1     070  06  06    129  09  09
012  01  1     071  65  ×     130  65  ×
013  06  6     072  01  1     131  05  5
014  03  3     073  02  2     132  75  -
015  03  3     074  93  .     133  43 RCL
016  03  3     075  01  1     134  08  08
017  02  2     076  09  9     135  85  +
018  69 □P     077  00  0     136  04  4
019  02  02    078  07  7     137  95  =
020  03  3     079  04  4     138  65  ×
021  06  6     080  09  9     139  02  2
022  02  2     081  75  -     140  95  =
023  04  4     082  01  1     141  22 INV
024  03  3     083  93  .     142  28 LOG
025  07  7     084  00  0     143  65  ×
026  02  2     085  07  7     144  03  3
027  04  4     086  06  6     145  00  0
028  03  3     087  52 EE     146  95  =
029  02  2     088  01  1     147  52 EE
030  69 □P     089  02  2     148  22 INV
031  03  03    090  94 +/-    149  52 EE
032  03  3     091  65  ×     150  42 STO
033  01  1     092  43 RCL    151  10  10
034  00  0     093  06  06    152  06  6
035  00  0     094  33 X²     153  65  ×
036  01  1     095  85  +     154  53  (
037  03  3     096  03  3     155  43 RCL
038  03  3     097  05  5     156  07  07
039  00  0     098  00  0     157  39 COS
040  07  7     099  93  .     158  75  -
041  01  1     100  07  7     159  01  1
042  69 □P     101  04  4     160  54  )
043  04  04    102  95  =     161  85  +
044  69 □P     103  22 INV    162  01  1
045  05  05    104  52 EE     163  03  3
046  98 ADV    105  42 STO    164  95  =
047  01  1     106  07  07    165  52 EE
048  00  0     107  38 SIN    166  22 INV
049  00  0     108  75  -     167  52 EE
050  93  .     109  01  1     168  22 INV
051  01  1     110  95  =     169  58 FIX
052  09  9     111  94 +/-    170  42 STO
053  36 PGM    112  65  ×     171  11  11
054  20  20    113  09  9     172  01  1
055  11  A     114  85  +     173  03  3
056  91 R/S    115  05  5     174  42 STO
057  58 FIX    116  95  =     175  00  00
058  04  04    117  58 FIX    176  69 □P
```

Beispiele

```
 MONDPOSITION AM?          MONDPOSITION AM?

1009.1980                 1023.1980

     S↑                        S↑

      M                         M

Neumond                                  Vollmond

      E                         E

                                M

 MONDPOSITION AM?          MONDPOSITION AM?

1014.1980                 1026.1980

     S↑                        S↑

M

      E                         E

                                M
```

Karl Simon

Ewiger Kalender auf dem PC 1211

Beschreibung

Bis auf den letzten Speicherplatz füllt der ewige Kalender den Speicher des PC 1211.

Berechnet werden kann der Julianische Tag (fortlaufende Tageszählung nach J. Scaliger, wie sie in der Astronomie und der Geschichtsforschung verwendet wird) im julianischen und im gregorianischen Kalender, der Wochentag im julianischen und gregorianischen Kalender, sowie die Differenz in Tagen zwischen zwei Daten im gregorianischen Kalender. Zum Schluß kann auch noch der Biorhythmus berechnet werden. Es gibt keine Datumsbeschränkung, wie sie die meisten anderen Kalenderprogramme haben. Es können also auch Daten aus der vorchristlichen Zeitrechnung (Eingabe negative Jahreszahl) verarbeitet werden. Zu beachten ist dabei, daß das Programm mit der astronomischen Jahreszählung (Jahr Ø) arbeitet. Die historische Jahreszählung kennt kein Jahr Ø, es muß also bei Daten vor Chr. Geb. ein Jahr zugezählt werden. Z.B.: −4712 astronomisch = 4713 v. Chr. historisch, Ø astron. = 1 v. Chr.

Das Programm wird im DEF-Modus mit SHIFT „A" gestartet und ist selbsterklärend. Es wird ein Menu angeboten, von dem man die gewünschte Berechnung auswählt. Jede Eingabe muß mit ENTER abgeschlossen werden, auch ein Weitergehen im Programm wird mit ENTER gestartet.

Hinweise:

Das Zeichen „§" steht für das Dollarzeichen. Bei der Auflistung wurden, der besseren Lesbarkeit wegen, Zwischenräume gesetzt. Beim Eingeben in den Computer sind diese wegzulassen, da das Programm den ganzen Speicher füllt!

Fortlaufende Tageszählung (nach J. Scaliger) JDjul Ø = −4712 Ø1 Ø1 (astronom. Zählung)

(bitte umblättern)

```
PROGRAMM:
 1Ø "Z" BEEP 1
 3Ø INPUT " JAHR= ";J," MONAT= ";M," TAG= ";T
 35 GOTO "PRUEF"
 37 IF V=1 THEN "Z"
 4Ø A=((J+4712)*36525-25)/1ØØ
 5Ø B=A-INT A
 6Ø C=INT A: IF SGN(B)=(-1)LET C=C-1
 7Ø D=(M-1)*31: IF M>=3LET D=(D-3-INT((M-2)*Ø.4))
 8Ø E=C+D+T
 9Ø F=E: IF (J/4-INT(J/4)=Ø THEN 11Ø
1ØØ GOTO 13Ø
11Ø IF M<=2 THEN 1ØØ
12Ø F=F+1: GOTO 1ØØ
13Ø K=(INT((J+7999)/1ØØ))-(INT((J+7999)/4ØØ))-62
14Ø G=F-K: IF (J/4ØØ-INT(J/4ØØ))=Ø THEN 16Ø
15Ø IF SGN F=-1 LET F=1ØØØØØØØ+F
151 IF SGN G=-1 LET G=1ØØØØØØØ+G
152 RETURN
16Ø IF M<2 THEN 15Ø
17Ø G=G+1: GOTO 15Ø

2ØØ "PRUEF" V=Ø
2Ø5 IF M<1 THEN 225
21Ø IF M>12 THEN 225
215 IF T<1 THEN 23Ø
22Ø IF T>31 THEN 23Ø
223 GOTO 37
225 PAUSE" MONAT FALSCH!": V=1: GOTO 37
23Ø PAUSE" TAG FALSCH!": V=1: GOTO 37

25Ø "Y" BEEP 1: INPUT" NOCHEINMAL J/N? ";U§
251 IF U§="N" BEEP3: PAUSE" AUF WIEDERSEHEN!": END
252 IF U§="J" GOTO "A"
253 GOTO "Y"

PROGRAMM: EWIGER KALENDER (Fortsetzung)

26Ø "A" CLEAR: BEEP 2
261 PAUSE" KALENDER"
262 PAUSE" JUL. TAG JUL.= B"
263 PAUSE" JUL. TAG GREG.= C"
264 PAUSE" WOCHENTAG JUL.= D"
265 PAUSE" WOCHENTAG GREG.= E"
266 PAUSE" DIFF. ZW. 2 DATEN= F"
267 PAUSE" BIORHYTMUS=G"
268 BEEP 1: INPUT" B,C,D,E,F,G? ";Z§: GOTO Z§

28Ø "B" GOSUB "Z"
285 PRINT" JD JUL.= ";F: GOTO "Y"
29Ø "C" GOSUB "Z"
295 PRINT" JD GREG.= ";G: GOTO"Y"
3ØØ "D" PAUSE" WOCHENTAG JUL.": GOSUB "Z"
3Ø5 Y=F: GOTO "TAG"
315 "E" PAUSE" WOCHENT. GREG.": GOSUB "Z"
32Ø Y=G: GOTO "TAG"
33Ø "F" PAUSE" DIFFERENZ": PAUSE" 1. DATUM": GOSUB "Z"
335 H=G: PAUSE" 2. DATUM": GOSUB "Z"
34Ø I=ABS(G-H): PRINT" DIFF.= ";I: GOTO "Y"
35Ø "G" PAUSE" BIORHYTMUS": GOTO "BIO"
36Ø "TAG" X=Y/7: X=X-INT X: X=X*7: X=361+X: X=INT(X+Ø.5): GOTO X
361 PRINT" MONTAG": GOTO "Y"
362 PRINT" DIENSTAG": GOTO "Y"
363 PRINT" MITTWOCH": GOTO "Y"
364 PRINT" DONNERSTAG": GOTO "Y"
365 PRINT" FREITAG": GOTO "Y"
366 PRINT" SAMSTAG": GOTO "Y"
367 PRINT" SONNTAG": GOTO "Y"

38Ø "BIO" PAUSE" GEBURTSDATUM": GOSUB "Z"
385 H=G: PAUSE" ZIELDATUM": GOSUB "Z"
39Ø I=ABS(G-H): O=SIN(I/23*36Ø): P=SIN(I/28*36Ø): Q=SIN(I/33*36Ø)
4ØØ PRINT" PHYS.= "; USING"##.##";O
4Ø5 PRINT" EMOT.= ";P
41Ø PRINT" GEIST.= ";Q:USING: GOTO "Y"
```

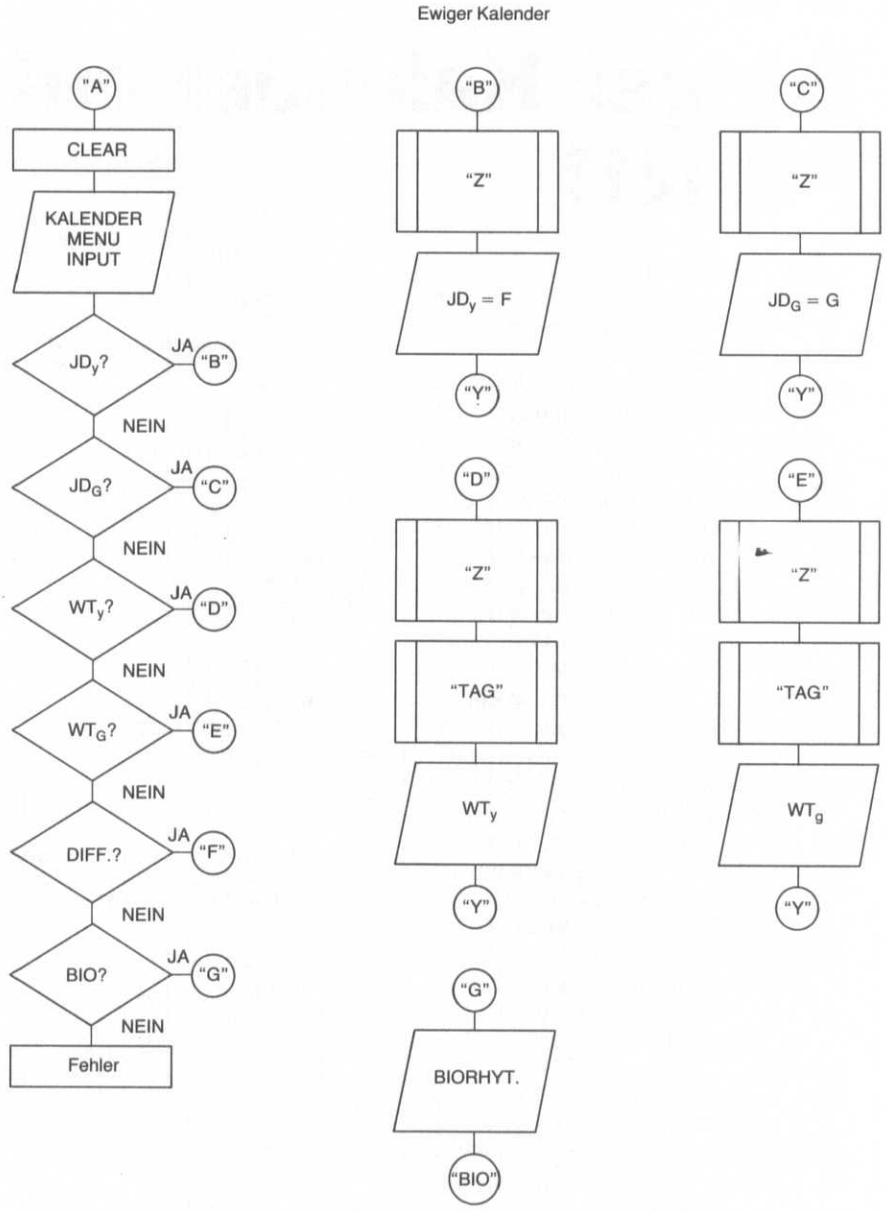
Ewiger Kalender auf dem PC 1211

Formeln:

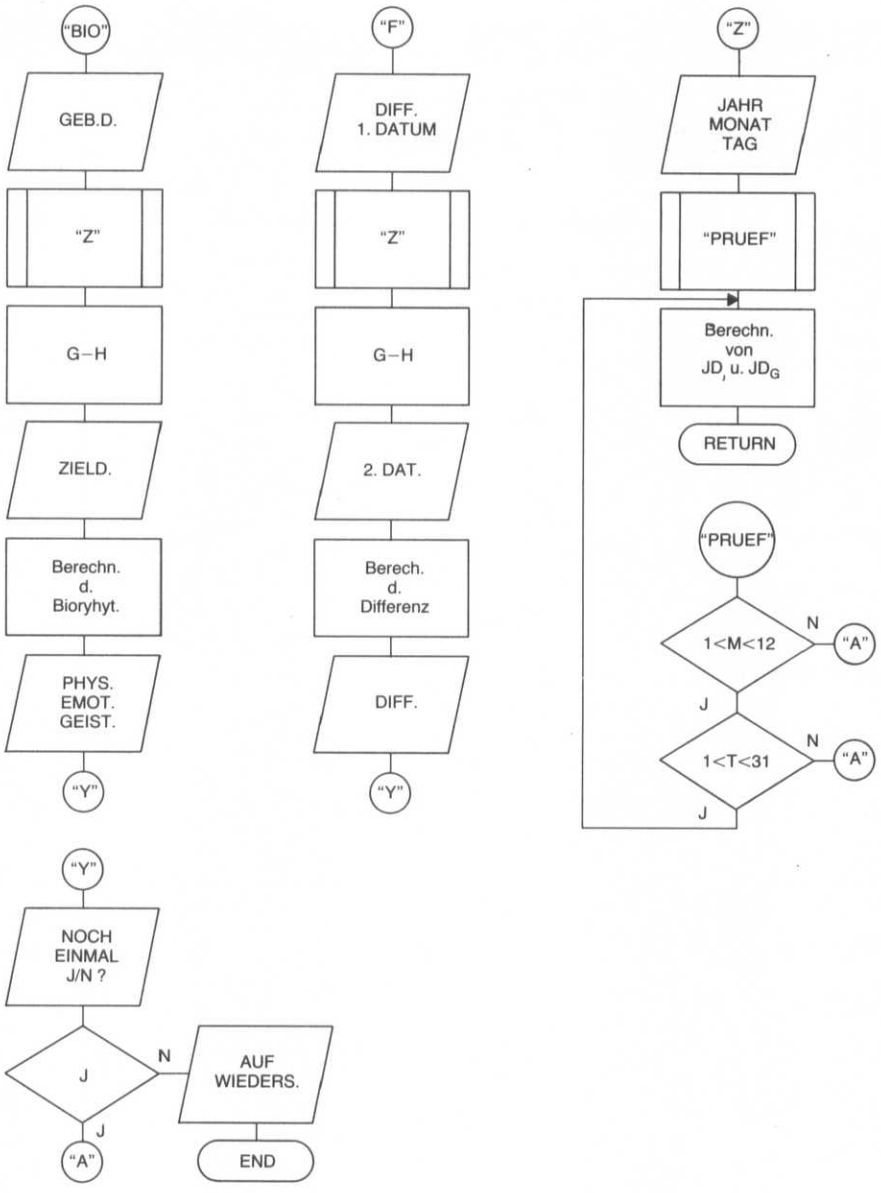
A = ((J+4712)×36525)/100
B = FRAC(A) FRAC einer negat. Zahl ist negativ
JDJ = INT(A) Wenn B neg., nimm JDJ−1 statt JDJ
JDM = (M−1)×31 Wenn M größer od. gleich 3, nimm JDM−3−INT((M−2) × 0.4) statt JDM
JDJMT = JDJ+JDM+T
JDJUL = JDJMT Wenn J julian. Schaltjahr u. M größer 2, nimm JDJUL+1 statt JDJUL
KORR = INT((J+7999)/100)− INT((J+7999)/400)−62
JDGREG = JDJMT − KORR Wenn J greg. Schaltjahr u. M größer 2, nimm JDGREG+1 statt JDGREG
WTJUL = Rest JDJUL/7 0 = Montag, 1 = Dienstag ...
WTGREG = Rest JDGREG/7 ··· 6 = Sonntag.

Statt negativem JD wird die dekadische Ergänzung gegeben.

	LABELS:
260	„A“ = Start
280	„B“ = JDjul
290	„C“ = JDgreg
300	„D“ = WTjul
315	„E“ = WTgreg
330	„F“ = Diff. zw. 2 Daten
350	„G“ = Biorhythmus
10	„Z“ = Berechnung JD
250	„Y“ = Ende − Abfrage
360	„TAG“ = Wochentagsber.
380	„BIO“ = Biorhythmusber.



Ewiger Kalender



Jürgen Twele

Berechnung des Pascalschen Dreiecks auf TI-59

Programmbeschreibung

Das Programm dient der Erstellung eines Pascalschen Dreiecks, wie es innerhalb des binomschen Lehrsatzes und zahlreicher anderer mathematischer Probleme erforderlich ist.
Natürlich kann das Dreieck auf dem Thermodrucker des TI-59 nicht waagrecht dargestellt werden. Es folgen die einzelnen Zahlenwerte also untereinander, zeilenweise voneinander durch eine Leerzeile getrennt.
Vor Ablauf des Programms müssen alle Datenregister und das Testregister Null zum Inhalt haben.
Der Programmstart erfolgt durch [A].

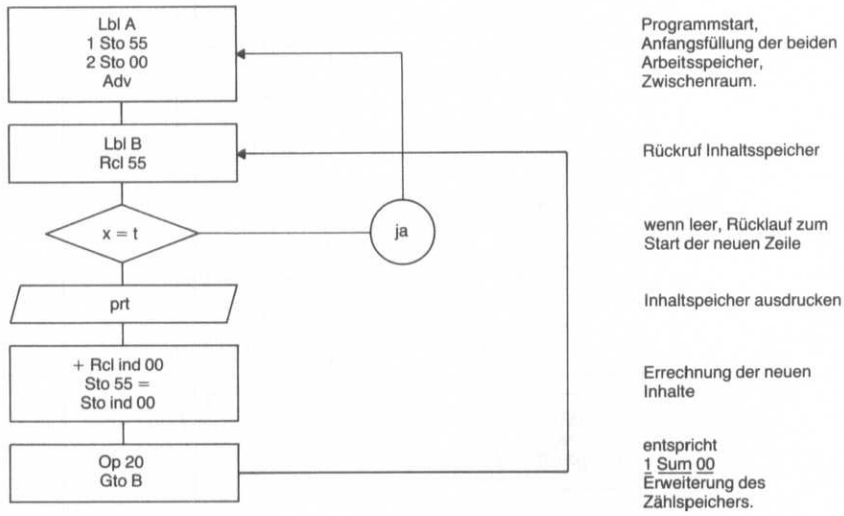


Bild 1

Programm - Start

```
001  11  A
010  12  B

000  76 LBL
001  11  A
002  01  1
003  42 STO
004  55  55
005  02  2
006  42 STO
007  00  00
008  98 ADV
009  76 LBL
010  12  B
011  43 RCL
012  55  55
013  67  EQ
014  11  A
015  99 PRT
016  85
017  73 RC*
018  00  00
019  42 STO
020  55  55
021  95  =
022  72 ST*
023  00  00
024  69 OP
025  20  20
026  12  B
027  00  0
028  00  0
029  00  0
030  00  0
031  00  0
```

Bild 2: Programmstart Pascalsches Dreieck

```
1.
1.
1.
1.
2.
1.
1.
3.
3.
1.
1.
4.
6.
4.
1.
1.
5.
10.
10.
5.
1.
```

```
1.
6.
15.
20.
15.
6.
1.
1.
7.
21.
35.
35.
21.
7.
1.
1.
8.
28.
56.
70.
56.
28.
8.
1.
```

```
1.
9.
36.
84.
126.
126.
84.
36.
9.
1.
1.
10.
45.
120.
210.
252.
210.
120.
45.
10.
1.
```

```
1.
11.
55.
165.
330.
462.
462.
330.
165.
55.
11.
1.
```

Bild 3

CROMEMCO VON COMICRO

nicht aus Prestige, sondern aus dem verständlichen Bedürfnis nach höchstem Leistungs-Preis-Verhältnis und optimaler Systembetreuung.

Anwendungssoftware
- Textverarbeitung
- Adreßverwaltung
- Finanzbuchhaltung
- Materialwirtschaft
- Lohn + Gehalt
- Architektur – Software
- Stundenplanoptimierung

Systemsoftware
- CROMIX, CDOS, CP/M
- BASIC, Structured BASIC
- FORTRAN, Rational FORTRAN
- COBOL, RPG II, LISP
- Macro-Assembler
- DBMS
- SDI-Graphics

mit Einzelplatz- und Mehrbenutzersystem, mit Mini-, Standard- und Festkopfplatten

COMICRO Deutschland GmbH
Mikrocomputersyteme, Beratung, Entwicklung und Vertrieb
Westring 105, D-4154 Tönisvorst, Tel. (0 21 51) 79 55 77

NEUHEIT für COMMODORE-COMPUTER

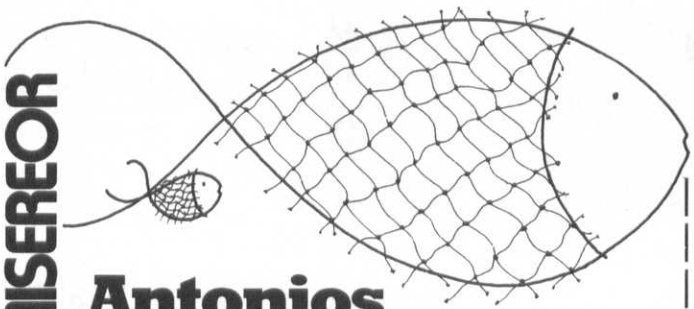
3000er Serie
4000er Serie
8000er Serie

Soft:
BASIC-COMPILER
1356,– DM inkl. MwSt.

Andere Produkte:
CBM-COMPUTER,
Floppy,
Drucker, Olivetti-Interface,
Computhink-Floppy,
A/D-Wandler.
Hochauflösende Graphik mit 64 000 Bildpunkten
u.v.a.m.

Hard:
UPGRADE-SYSTEM
3064, 4064 und 8064
Erweitern Sie Ihren CBM um 64 KByte (oder 128 KByte) auf 96 KByte (für Programme und Daten!)
ab 2599,– DM inkl. MwSt.
Händleranfragen erwünscht

Info durch
SPIMA·COMPUTER·GMBH
Turbinenstraße 4 · D-6800 Mannheim 31 · Tel. (06 21) 72 15 15 · Telex 04 63 708 spimad



MISEREOR

Antonios Netz ist kleiner.

„Gib einem Hungernden einen Fisch, und du machst ihn satt. Schenke ihm ein Netz, und er wird nicht mehr hungern."
Das ist einleuchtend, es sei denn:
– deine Netze sind größer
– deine Schiffe sind schneller
– deine Fischhändler sind gerissener
deine Fischer sind erbarmungsloser als Antonio.
Alle Anstrengungen der Entwicklungshilfe laufen ins Leere, wenn wir den Menschen der Dritten Welt keine fairen Chancen einräumen.

ANDERS LEBEN: GEMEINSAM HANDELN
Spendenkonto 556-505 Postscheckamt Köln
Mozartstraße 9, 5100 Aachen

PET/CBM Besitzer

Kennen Sie **SYNTAX** – das Programm-Magazin auf Kassette?

Es bringt jeden Monat 5 neue Programme in deutscher Sprache aus allen Bereichen. Zum Beispiel Datei-Systeme, Textverarbeitung, Lehrgang Maschinen-Sprache, User-Programme etc.

Kenner der **SYNTAX-MAGAZINE** loben Leistung und Preis.

Fordern Sie gleich heute noch kostenlose Informationen von

SYNTAX
Soft- u. Hardware GmbH
P. B. 1609, 7550 Rastatt
Tel. 0 72 22/3 42 96

INSO
INDIVIDUELLE SOFTWARE ENTWICKLUNG

Jürgen Ehrmann
Tel. (0 70 71) 3 46 73
Breslauer Straße 30
7400 Tübingen

Wir bieten **exakte Software-Problemlösungen** für Minis und Micros.

Ebenso führen wir flexible Standard-Software, z.B.

- Superdateiverwaltung (10–20-zeilig, wahlfreies Sortieren, Heraussuchen, Listen usw.)
- Optionen (Textverarbeitung oder Kontoführung)
- Bestands-/Inventurlistenführung
- Adressenverwaltung für Versicherungen (Standardbriefe, Merkblätter, Bankeinzugsbelege)
- Finanzbuchhaltung

Wir beraten Sie auch gerne (und unabhängig von Händlerinteressen) bei der Hardware-Wahl.

Rufen Sie einfach an oder fordern Sie **unverbindl. Info** (2,– DM Schutzgebühr) an!

commodore
mca Software

mca – **Lohn und Gehalt**

mca – **Textsystem mit Adreßverwaltung/ dt. Tastatur**

PASCAL – Compiler für cbm 8001 – (einschl. dt. Handbuch)

Zu beziehen über den Commodore-Fachhandel oder

mca Software
Gesellschaft für Micro-computeranwendungen mbH
7120 Bietigheim
Telefon (0 71 42) 5 10 90 – 5 10 99

DEUTSCHLAND

Aachen

RMI
Nachrichtentechnik
Gesellschaft mit
beschränkter Haftung

EXO COMPUTER/CALIFORNIA COMPUTER SYSTEMS/FMG/MICROSOFT
1. Aachener Computerladen
Rosstraße 7, Tel. 02 41 /2 40 70, Telex 8 325 21 rmi d

Aschaffenburg

CSG **Das Computerzentrum im Untermaingebiet**
Computer Service GmbH
Frohsinnstr. 32, 8750 Aschaffenburg, Tel. (0 60 21) 1 42 12

Augsburg

Computer-Markt · Heilbronn · Computer-Markt · Heilbronn
lucky dates
Verkaufsstützpunkt Augsburg:
Radio Lukas, Schutterstr. 2
8906 Gersthofen/Batzenhofen
Telefon (0 82 30) 95 55
Computer-Markt · Heilbronn · Computer-Markt · Heilbronn

Bad Honnef

ABC 80
Der professionelle Klein-Computer
Software – Zubehör
id **industrie design**
5340 Bad Honnef 6
Himberger Straße 5a
Telefon (0 22 24) 8 01 26

Berlin

Sharp MZ 80, Sorcerer, Commodore, Oki Drucker, Tally Mannesmann, Plotter, Digitizer, Floppy Disk, Plattenlaufwerke mit und ohne Controller.
Preiskatalog auf Anfrage!
Berkaer Str. 39, 1000 Berlin 33, Tel.:030/826 16 10
KOMO ELEKTRONIK GMBH

DEC, Epson, Facit, Lear Siegler, Mannesmann, Qume, Teletype, Tele Video.
Computer-Peripherie, Meßtechnik
PK elektronik Poppe GmbH
Lietzenburger Str. 91, 1000 Berlin 15, Tel. (030) 883 10 58
PK elektronik

Bielefeld

iBS COMPUTERTECHNIK
Artur-Ladebeck-Str. 143 — **Computersysteme**
4800 Bielefeld 14 — **Hardwareentwicklung**
Telefon 05 21/15 28 07 — **Softwareentwicklung**

Bonn

APEX SCS
SATELLITE COMPUTER SYSTEMS
MICRO-COMPUTER
KOMPLETTSYSTEM
HARDWARE/SOFTWARE
SERVICE
für Selbständige, Klein- und Mittelbetriebe

SYNERGY GmbH
Pfarrer-Hürth-Str. 67, 5300 Bonn
Tel. 0228/213480

Bremen

BREMENS 1. MICROCOMPUTERSHOP
Systeme-Hardware-Software-Peripherie
Lektüre – BERATUNG – ENTWICKLUNG

Hans Schröder, Tel. (04 21) 45 97 79
Föhrenstraße 19, 2800 **BREMEN** 1

WEBER Funk
Funk – Elektronik – Computer – Video
Emil-von-Behring-Straße 6
Telefon (04 21) 49 00 10/19

APEX SCS
SATELLITE COMPUTER SYSTEMS
MICRO-COMPUTER
KOMPLETTSYSTEM
HARDWARE/SOFTWARE
SERVICE
für Selbständige, Klein- und Mittelbetriebe

ISI Electronic Handels GmbH
Föhrenstraße 19, 2800 Bremen 1
Tel. 0421/459779

Darmstadt

micro Computer-Zentrum
Die **Nr. 1** in Hessen
Alsfelder Str. 7 · 6100 Darmstadt
am Meßplatz · Telefon 06151/76032

Dortmund

city-elektronik
Bauteile- Funk- und Meßgeräte
APPLE, ITT-2020, CBM, SHARP, EG-3003
4600 DORTMUND 1
Güntherstr. 75 + Weißenburger Str. 43
Telefon 02 31 / 57 22 84

Düsseldorf

DER COMPUTERLADEN MIT DEM KOMPLETTEN EDV-ANGEBOT
DATA BECKER GMBH
im Hause Auto Becker
Merowingerstr. 30 · 4000 Düsseldorf
Tel. 0211/312085/86 · Telex 08582874

Duisburg

KIRCHNER-ELEKTRONIK-DUISBURG
DIPL.-ING. ANTON KIRCHNER
4100 Duisburg-Neudorf, Grabenstr. 90, Tel. 37 21 28, Telex 08 55 531

Essen

HÜLSEWIG COMPUTER SYSTEME
APPLE · CBM · VIDEO-GENIE · TANDY · DAI · SUPERBRAIN · EPSON · OLYMPIA · CENTRONICS · PLOTTER · MONITORE · BUSINESS-SYSTEME · INTERFACES · MARKEN-DISKETTEN · SOFTWARE.
Händleranfragen erwünscht.
Am Wünnesberg 9, 4300 Essen 1, Tel. (02 01) 71 39 04

Frankfurt

» COMPUTERHAUS «
Hardware · Software · Beratung–Service
apple Honeywell Bull
G + B GmbH · 6 Ffm. · Tel. (06 11) 44 60 16
Gaußstr. 12 (Nähe Zeil) eigene P

Henneveld KG
Moderne Büro-Organisation
3x im Rhein-Main-Gebiet
Frankfurt Mainz Wiesbaden
Der autorisierte Vertragshändler von HEWLETT-PACKARD u. COMMODORE
Zeil 127 (Hauptwache) · 6000 Ffm. · ☎ (0611) 28 14 82

micro Computer-Zentrum
Die Nr. 1 in Hessen
Dreieichstr. 59 · am Lokalbahnhof
6000 Frankfurt 70 · Tel. 0611/625048

APEX SCS
SATELLITE COMPUTER SYSTEMS
MICRO-COMPUTER KOMPLETTSYSTEM HARDWARE/SOFTWARE SERVICE für Selbständige, Klein- und Mittelbetriebe
Sven Computer GmbH
Bleichenbrücke 10, 2000 Hamburg 36
Tel. 040/34 38 26

Koblenz
"bhg... Computersysteme
commodore
apple II
Compucorp
– Hardware – Peripherie
– Beratung – Service
– Software – Systemanalyse
– Entwicklungssysteme
Moselring 1 · 5400 Koblenz · Tel. (02 61) 1 24 88

Köln
Colonia Computer
apple II
und andere Systeme
Centronics
Trendcom
Binder
PASCAL, FORTH, APPLE PIE
Kundenspezifische Software Entwicklung
Colonia Computer GmbH
Lindenstr. 73 · 5000 Köln 1 · 0221/215736

Freiburg
a electronic gmbh
Telex: 07 72 642 aaa-d
Habsburgerstr. 134
78 FREIBURG, Tel. (07 61) 27 68 64
Bauelemente – Bausätze – µP's
Meßgeräte – Zubehör – Fachliteratur
Fachgeschäft für Elektronik und µ-Computer

Hannover
MARENO
Datensysteme
3000 Hannover – Georg-Str. 20
Tel. (05 11) 1 27 18

Fachgeschäft für:
antennen, funkgeräte, bauteile, computer und zubehör
IN KÖLN UND BONN
P+M elektronik
5 KÖLN 80 BUCHHEIMERSTR 19
5 KÖLN 1 AACHENERSTR 27
53 BONN STERNSTR. 102

Göttingen
Computerdienst
Brockhaus & Müller KG
Wir vertreiben
Apple
Commodore
Weender Landstr. 3 · 3400 Göttingen
(IDUNA-ZENTRUM) 05 51/5 52 82

Heilbronn
Computer-Markt · Heilbronn · Computer-Markt · Heilbronn
lucky dates
Wir setzen Maßstäbe in der Beratung
Am Wollhaus 1, 7100 Heilbronn,
Tel. (0 71 31) 6 93 33. Telex 7 28 273

Krefeld
SVI
Computersysteme
Commodore, Hewlett Packard, Sharp, apple®
Peripherie
Farb-Terminals, SW-Terminals, Matrixdrucker, Typenraddrucker, Plotter, Meßgeräte
4150 Krefeld, Breuershofstr. 40
Tel. (0 21 51) 3 60 56
Software-Vertrieb-Informationsverarbeitung

Hamburg
alfred graumann
Elektronik-Vertrieb
(0 40) 2 50 30 50 · Grevenweg 83 · 2000 Hamburg 26

Herrenberg
HARTNAGEL MICRO-COMPUTER & HARTNAGEL ELEKTRONIK-RECHNER
Texas-Instruments-Rechner zu absoluten Tiefstpreisen
Hardware und Software für kommerzielle Anwendung
Programmierung Anwendungsberatung Organisation
Die kompl. Problemlösung – »Alles aus einer Hand«
Versandhandel: alle führenden Hersteller
Postfach 1102 · D-7033 Herrenberg
Tel.: 07032/32379 u. 26050 · Telex 7-265876 htgld

Lahr
LEHMANN electronic
Langenwinkler Hauptstr. 5, 7630 LAHR
9–12 und 14–18 Uhr, Tel. (0 78 21) 49 90
NORTH-STAR, SHARP, VIDEO GENIE, EXIDY SORCERER, HP-85, HP-PLOTTER, EPSON, NEC, SOROC, TVI, DRUCKER

RF MRG
RAMELOW Informations- und Communications-Technik GmbH
Biernatzkistraße 28, 2000 Hamburg 50
Tel. 040/3 80 93 54
Hardware · Software · Entwicklung und Vertrieb
Mehr als 10 Jahre Erfahrung in der EDV-Beratung
Hardware-Interessenten fordern bitte Produktinformation an!
Betriebssystem CP/M Compatibel OASIS Schwerpunkt Z 80 (A)-Systeme

Kassel
Computerdienst
Brockhaus & Müller KG
Das APPLE-Service-Zentrum für Nord-Hessen
Triftstraße 27 · 3502 Kassel-Vellmar
(05 61) 82 64 47

Landau/Pfalz
DATA-SERVICE GmbH DATAS
Computer-Systeme · Software-Entwicklug · EDV-Service
● Sorcerer · Apple · CBM · System 10
● Matrix- und Typenrad-Drucker
● Individuelle Software-Entwicklung
● Komplett-Lösungen
6740 Landau/Pfalz, Kramstraße 23
Telefon (0 63 41) 8 45 77 und 2 07 29

Lübeck

Sharp
apple
VIDEO GENIE
Zubehör, Software, Service

multitron

Jessen & Lenz
Hinter der Burg 1, 2400 Lübeck
Telefon (04 51) 70 50 30/70 51 51

Mainz

Henneveld KG Moderne Büro-Organisation

3x im Rhein-Main-Gebiet Frankfurt Mainz Wiesbaden

Der autorisierte Vertragshändler von HEWLETT-PACKARD u. COMMODORE

Hechtsheimer Str. 121 · 6500 Mainz · ☎ (0 61 31) 8 50 91

München

Praxisgerechte Komplettlösungen von Ihrem zuverlässigen EDV-Partner

MC MK Micro-Computertechnik
Manfred Krause Ing. (grad.)

Inderstorferstr. 59 · Postf. 21 04 07 · 8000 München 21 · Tel. (0 89) 58 10 06

ludwig

Microcomputer

apple computer

Hardware – Peripherie – Software

Apple II ab sofort mit Groß-, Kleinschreibung und Umlauten – alle ASCII – Zeichen

TEXAS INSTRUMENTS
Home Computer System TI 99/4-PAL

Neu: An Ihr PAL-TV-Gerät anschließbar
Jetzt noch mehr Software lieferbar

Beratung — Verkauf — Kunden-dienst
Ingolstädter Str. 62L (Euro-Industrie-Park)
8000 München 45, Fernruf (0 89) 3 11 10 15

MMOS UNTERNEHMENSBERATUNG COMPUTER-$HOP TRAINING----------SERVICE
Marketing And Management Operation Service

FÜR INDUSTRIE + HANDEL + PRIVAT
Wir beraten Sie bei Ihrer EDV-Lösung.
Sie sehen im Demonstrations-Raum:
APPLE mit dem starken Betriebssystem,
mit der starken Graphik,
mit kompletter Buchhaltung,
mit Angebotssoftware,
WEITERE BRANCHEN-PAKETE GESUCHT
CP/M-Systeme mit breiter Software.
LSI-11/DEC-Systeme mit Profi-Booster.
MAI-kompatible Systeme für hohe Leistung mit viel kommerzieller Software im Markt.
MONTAGS-SEMINARE + EINFÜHRUNG
200 m ab Ausfahrt Oberhaching; Parken!
WENDELSTEINWEG 6/TÖLZER STR.
8028 TAUFKIRCHEN/MÜNCHEN
ORTSTEIL POTZHAM
TELEFON (0 89) 6 12 61 22

Münster

AIM-65 (Rockwell)

apple computer

Reichhaltiges Zubehör
Epson Gebietsvertretung
Schnellversand innerhalb 24 Stunden

Norbert Hunstig
Labor für Nachrichtentechnik
Olfersstraße 3-5 (Nähe Dahlweg)
D-4400 Münster i. Westf.
Tel.: (02 51) 7 63 48 · Telex: 8 92 496 hunms d

apple computer

SE-System-Elektronik
Hammer Str. 223, 4400 Münster Tel. (02 51) 79 69 29, Telex 8 91 464

Neuss

SOCOMP
Microcomputer-Handelsgesellschaft mbH

Microcomputer	*4040 Neuss*
Softwareberatung	*Rheinwallgraben 28*
Peripherie	*Telefon (0 21 01) 2 57 89*
Service	*Telex 08 517 973*

Nürnberg

apple computer

bei: SOLARIM

Solartechnische Gesellschaft m. b. H.
Roritzer Str. 28, Tel. 0911 / 33 48 35
8500 NÜRNBERG 90, Postf. 910 349
apple Vertragshändler Nordbayern

Oldenburg

apple computer

SYNTAX EDV-Beratung und Microcomputervertrieb GmbH

Ammerländer Heerstraße 56, 2900 Oldenburg
Telefon 04 41 / 7 62 06

Osnabrück

Heinicke-electronic

Apple · Tandy · Sharp · Videogenie · Centronics

Kommenderiestr. 120 · 4500 Osnabrück · Tel. (05 41) 8 27 99

Paderborn

bcb computerbase
personal
commercial computersysteme
industrial

comuter – import/export · verkauf · beratung · entwicklung · produktion – computer

bcb · bussmann computerbase 4790 paderborn
hathumarstr. 16 · tel. (0 52 51) 2 31 35

Regensburg

Jodlbauer-Elektronik
Bauteile – Halbleiter – Geräte
Mikrocomputer
Woehrdstraße 7, Telefon 0941/57924
8400 Regensburg

Saarlouis

6630 Saarlouis,
Lothringerstraße 9

Minninger
ELEKTRONIK + FUNK

Stuttgart

Arlt Bauelemente für die Elektronik
Industrie- u. Laborbedarf

Arlt Elektronik, 7000 Stuttgart 1
Kathannenstraße 22, Fernruf 07 11/24 57 46

Stuttgart

MICRO-COMPUTER-STUDIO
Ihr Partner in
STUTTGART
für das gesamte
Programm von
APPLE · TANDY
SHARP
Informations-Prospekt gratis
eschenbrenner + sauer gmbh
Ludwigstr. 87 A · Stgt.-W. · Tel.: 0711/61 22 52

Siemens-PC-100
S-80, Video-Genie
Apple Plus
Olympia-Boss
Zilog MCZ
R. HALLER, M.A.
Fachgeschäft für Bürocomputer
Lehrmittel- und Buchhandlung
7000 STUTTGART 1 · Tel. (07 11) 28 15 23
Landhausstr. 21 (Kernerplatz)

IC: Intacom
Computer Elektronik
und Software
Vertrieb GmbH
Hohenheimer Str. 11 Postfach 10 03 23
D-7022 Leinfelden-Echterdingen 1
Telefon 07 11/75 10 59 Telex: 07 255 746

Computer:
Hardware-Peripherie-
Software

Ulm

computer + text
organisation systeme
gmbh
Anwendersoftwarelösungen
Seit Jahren einer der Marktführer im techn.-wissensch. Ingenieurwesen, sowie in der Textbe- und -verarbeitung.
Dialogintensive kommerz. Software ist unsere Stärke.
Lehrgänge
Für jedes unserer Systeme führen wir Anwender- und Programmierkurse im halbjährlichen Zyklus durch.
Service
Wartung unserer Systeme in eigener Werkstatt in Ulm.
Generalvertreter der Firmen
TRIUMPH-ADLER AG (alphatronic, dds-CS, BITSY, SE 2000),
COMMODORE-Büromaschinen (alle cbm-Modelle).
Eberhard-Finckh-Straße 34,
7900 Ulm-Böfingen, ☎ (07 31) 2 61 07.

wagner GmbH
COMPUTER VERTRIEB
Zubehör, Software, Service
AUSSTELLUNG und VERKAUF
Lachhausstraße 22,
7900 Ulm/Einsingen · Tel. (0 73 05) 80 81

VS-Villingen

Tandy Radio Shack
H.-U. Fischer & Partner
Gottlieb-Daimler-Str. 7
7730 VS-Villingen
Telefon (0 77 21) 7 00 46
Vertragshändler – Beratung Hardware Software Service

Wiesbaden

Henneveld KG Moderne Büro-Organisation
3x im Rhein-Main-Gebiet Frankfurt Mainz Wiesbaden
Der autorisierte Vertragshändler von HEWLETT-PACKARD u. COMMODORE
Bleichstr. 30 · 6200 Wiesbaden · ☎ (0 61 21) 30 70 91

Worms

as **electronic**
Renzstr. 39 Nähe Hbf
Telefon (0 62 41) 2 78 67
WORMS (bef)

Wuppertal

APEX SCS
SATELLITE COMPUTER SYSTEMS
MICRO-COMPUTER
KOMPLETTSYSTEM
HARDWARE/SOFTWARE
SERVICE
für Selbständige, Klein- und Mittelbetriebe

COMPACT-Mikrocomputer GmbH
Hofkamp 24, 5600 Wuppertal 1
Tel. 0202/44 09 03

ÖSTERREICH

GENERALVERTRETUNG
CHIP · Buchservice
Fachbuch Center Erb
Amerlingstraße 1 · 1061 Wien
Tel. 56 62 09, 57 94 98

Vorarlberg

COMPUTER SHOP WEST
Dipl. kfm. Kurt Herran
Lauteracherstr. 21 – **A-6922 Bregenz-Wolfurt**
Tel.: (0 55 74) 3 16 06
Telex: 5 7 693

SCHWEIZ

GENERALVERTRETUNG
CHIP · Buchservice
(Preise DM = sfr)
THALI AG
Fachliteratur, Bausätze, Bauteile
6285 Hitzkirch · Tel. 041/ 85 28 28

Söll, Wolfgang
Kirchner, Jörg-Hagen
Digitale Datenspeicher
Kamprath-Reihe kurz und bündig
220 Seiten, 251 Abbildungen
2farbig
38,– DM/ISBN 3-8023-0117-X

Band-, Platten-, floppy disk- und Bibliotheksspeicher als bekannte Datenspeicher. Holographischer, Bildplatten-, Magnetblasen-, Supraleitender Speicher sind technologisch ganz neu. Halbleiter-Schreib/Lese- und Festwertspeicher, PROM, Ladungsschiebespeicher und Sonderspeicher für Mikroprozessoren mit Anwendungsbeispielen. Für besondere Anwendungen: Fluid-, Mechanische und Elektromechanische Speicher

Fachbücher für Technik und Wirtschaft
Katalog liefert
VOGEL-VERLAG
Postfach 67 40
8700 Würzburg 1

Inserentenverzeichnis

BSP T. Krug 129

Colman 128
Comicro 181
Compusoftpak 131
Corvallis 171

Data-Becker 112

Ehrmann 181
Epson ..... 4. US

Intersoft 4, 13

mca 181
MK 133
Münzenloher 40

S + B 128
Schwind 129

Special Software Service 85
Spima 181
Syntax 181

Teach 13
Trommelschläger 171

Zander 17
Zimmermann 85

CHIP SPECIAL

Programme II

1981

ISBN 3-8023-0701-1

Redaktionsdirektor: Richard Kerler (verantwortlich für den Inhalt)

Redaktion: Gerhard Bader, Ruth Bosch

Gestaltung: Ludwig Lechler

Titelillustration: Ernst Jünger

Redaktionsservice: Marianne Weissbach

Redaktion: Vogel-Verlag KG Würzburg, Redaktion CHIP, Bavariaring 9, 8000 München 2, Telefon (0 89) 53 03 02, Telex 05 216 449, Telefax (0 89) 535 000

Verlag: Vogel-Verlag KG, Postfach 67 40, D-8700 Würzburg 1, Tel. (09 31) 41 02-1, Telex 06 8 883, Telefax (09 31) 41 02-5 29, Telegramme: CHIP-Würzburg

Verlagsleiter: Hugo E. Martin, Würzburg

Anzeigenleiter: Harald Kempf, Würzburg (verantwortlich für Anzeigen)

Anzeigenservice: Margit Ebert, Frauke Reith, Loretta Hamperl, Axel Winheim, Postfach 67 40, 8700 Würzburg 1, Telefon (09 31) 41 02-3 59, Telex 06 8 883

Anzeigenverkauf Bayern und Baden-Württemberg: Klaus von Biedermann, Verlagsvertretung CHIP, Bavariaring 9, 8000 München 2, Telefon (0 89) 53 35 36, Telex 05 216 449

Vertriebsleitung: Axel Herbschleb, Würzburg; Generalvertretung Österreich: Fachbuch Center Erb, Amerlingstr. 1, A-1061 Wien 6, Telefon (02 22) 56 62 09; Generalvertretung Schweiz: Thali AG, CH-6285 Hitzkirch, Telefon (0 41) 85 28 28.

Vertrieb Handelsauflage: Vereinigte Motor-Verlage GmbH, Leuschnerstr. 1, 7000 Stuttgart 1, Tel. (07 11) 20 43-1, Telex 07 22 036

Bezugsmöglichkeiten: Bestellungen nehmen der Verlag, die o.a. Generalvertretungen, jedes Postamt und alle Buchhandlungen im In- und Ausland entgegen. Sollte die Zeitschrift aus Gründen, die nicht vom Verlag zu vertreten sind, nicht geliefert werden können, besteht kein Anspruch auf Nachlieferung oder Erstattung vorausbezahlter Bezugsgelder.

Bankverbindungen: Bayerische Vereinsbank, Würzburg, Dresdner Bank AG, Würzburg, Postscheckkonto: Vogel-Verlag, Postscheckamt Nürnberg.

Gesamtherstellung und Versand: VOGEL-DRUCK WÜRZBURG, Max-Planck-Straße 7/9, D-8700 Würzburg.

Unverlangte Manuskripte werden nur zugesandt, wenn Rückporto beigefügt ist. — Für die mit Namen oder Signatur des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. — Die in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Nachdruck, Vervielfältigung sowie Speicherung in Datenverarbeitungsanlagen nur mit ausdrücklicher Genehmigung des Verlages. Jede im Bereich eines gewerblichen Unternehmens hergestellte oder benutzte Kopie dient gewerblichen Zwecken gem. § 54 (2) UrhG und verpflichtet zur Gebührenzahlung an die VG Wort, Abteilung Wissenschaft, Goethestraße 49, 8000 München 2, von der die Zahlungsmodalitäten zu erfragen sind.

Die Redaktion hat die Manuskripte, Bauanleitungen und Schaltungen sorgfältig geprüft. Für Fehler im Text, in Schaltbildern, Aufbauskizzen, Stücklisten usw., die zum Nichtfunktionieren oder evtl. zum Schadhaftwerden von Bauelementen führen, kann keine Haftung übernommen werden.

Sämtliche Veröffentlichungen in CHIP erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes, auch werden Warennamen ohne Gewährleistung einer freien Verwendung benützt.

CHIP – Kontaktkarte

Bitte mit 50 Pfennig freimachen

Absender

Vor- und Zuname

Beruf

Straße und Nr.

PLZ

Wohnort

CHIP

Zeitschrift für Microcomputertechnik

Ich habe in Ausgabe .... auf Seite .... Ihre Anzeige ☐ Ihre Produktbesprechung ☐ gelesen.
Ich möchte mehr über Ihr Produkt ..... erfahren.
Bitte senden Sie mir Informationsmaterial.
Ich bestelle unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen:

Menge	Produkt	à DM	gesamt DM

Für Jugendliche unter 18 Jahren ist die Unterschrift des Erziehungsberechtigten erforderlich.

Datum Unterschrift

CHIP-Buchservice

Bitte Absender nicht vergessen!

Ich bestelle mit Rechnung

Anzahl	Autor, Titel	Preis

zuzügl. 2,– DM Versandkostenanteil

Datum Unterschrift

CHIP-Abrufkarte

Ja, liefern Sie Chip ab Monat _____ für die Dauer eines Jahres und weiter bis zur Abbestellung* zum Jahresbezugspreis von 55,— DM** für 12 Hefte frei Haus. Ausland: 64,— DM**. Österreich und Schweiz: siehe Impressum.

Absender und Lieferanschrift

Bitte in jedes Feld nur einen Druckbuchstaben (ä = ae, ö = oe, ü = ue)

Vor- und Zuname

Beruf

Straße und Hausnummer

PLZ Wohnort

Datum/Unterschrift

* Abbestellung ist nach Ablauf der Mindestbezugszeit mit einer Kündigungsfrist von 2 Monaten jeweils zum Quartalsende möglich

** Gültig ab 1. 1. 1981

CHIP – Kontaktkarte

Bitte mit 50 Pfennig freimachen

Absender

Vor- und Zuname

Beruf

Straße und Nr.

PLZ

Wohnort

CHIP

Zeitschrift für Microcomputertechnik

Ich habe in Ausgabe .... auf Seite .... Ihre Anzeige ☐ Ihre Produktbesprechung ☐ gelesen.

Ich möchte mehr über Ihr Produkt ..... erfahren.

Bitte senden Sie mir Informationsmaterial.

Ich bestelle unter Anerkennung Ihrer Liefer- und Zahlungsbedingungen:

Menge	Produkt	à DM	gesamt DM

Für Jugendliche unter 18 Jahren ist die Unterschrift des Erziehungsberechtigten erforderlich.

Datum Unterschrift

CHIP Abrufkarte

Bitte freimachen

Antwort

CHIP
Leser-Service
Vogel-Verlag
Postfach 67 40

D-8700 Würzburg 1

CHIP Buchservice

Absender

Vor- und Zuname

Beruf

Straße und Nr.

Wohnort

PLZ

Bitte freimachen

Antwort

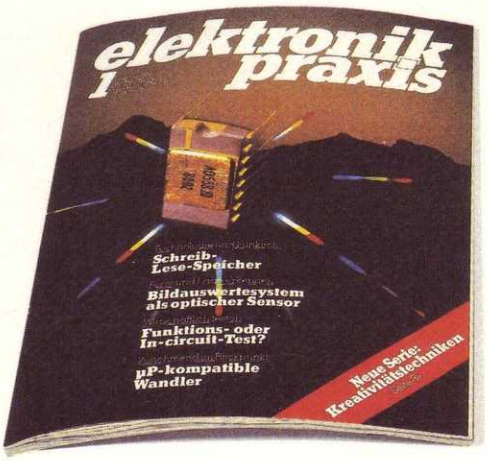
CHIP
Buchservice
Vogel-Verlag
Postfach 67 40

D-8700 Würzburg 1

Ein Bergsteiger muß sich allerhand zumuten, nur um den Überblick zu gewinnen. Die Leser der neuen »elektronikpraxis« bekommen ihn regelmäßig geliefert. Von einer Redaktion, die sich in der Welt der Elektronik auskennt und darüber berichtet: anwendungsbezogen, praxisorientiert und verständlich.

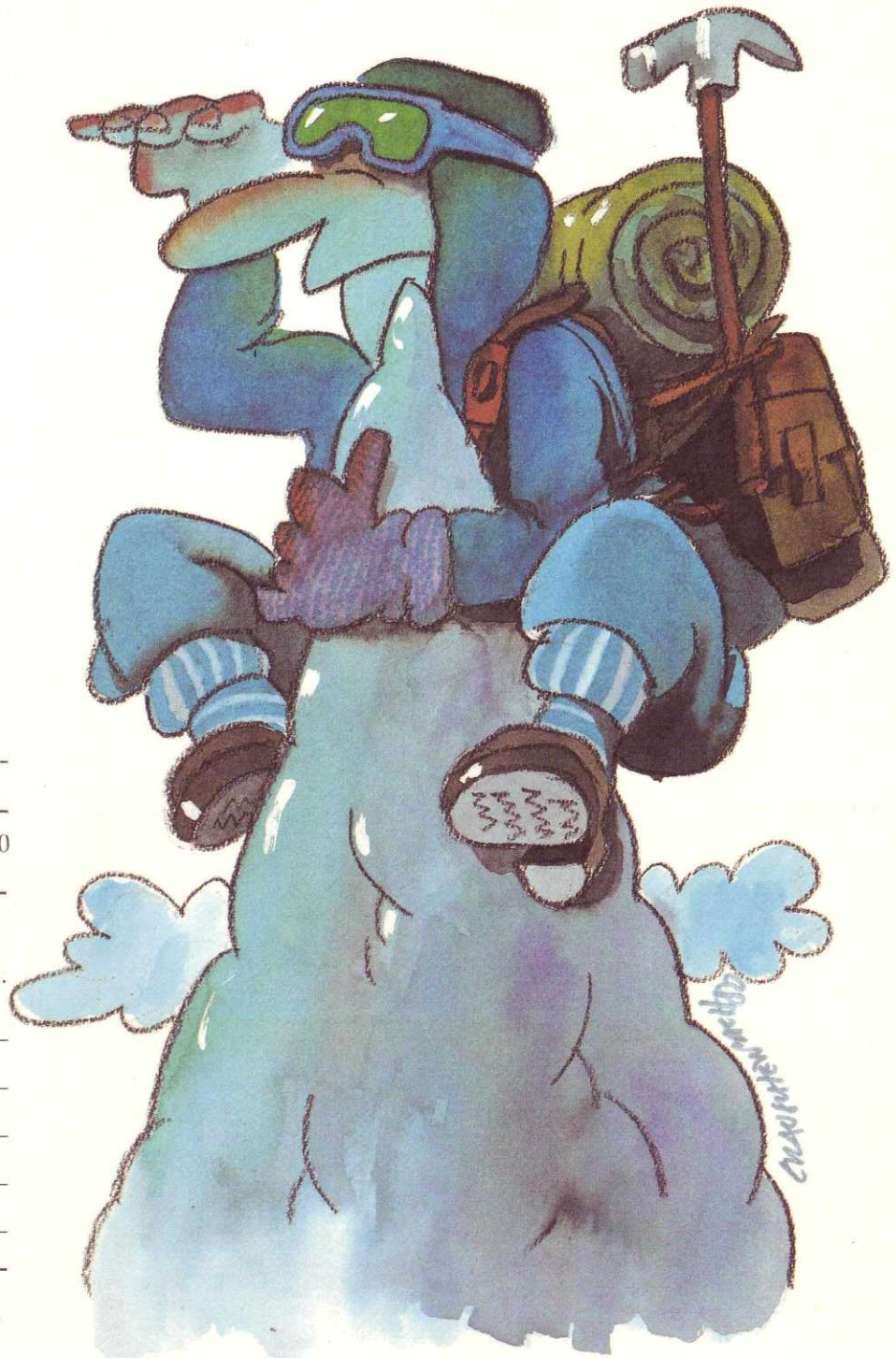
Und das bietet die neue »elektronikpraxis«

- **Grundlagenartikel über Technologien, Verfahren und Methoden.**
- **Applikationsberichte über die Anwendung von Bauelementen und Meßgeräten.**
- **Übersichtsartikel als Sammlung von Fakten mit didaktischer Kommentierung.**
- **Recherchierte Produktmeldungen, die aufzeigen, wie welches Produkt wo eingesetzt werden kann.**
- **Interviews, Firmenportraits und vieles andere aus der Welt der Elektronik.**



»elektronikpraxis«
eine Fachzeitschrift aus dem Vogel-Verlag

Postfach 67 40, D-8700 Würzburg 1
Telefon (09 31) 41 02-1, Telex 06 8 883



Vogel-Verlag, Abt. 262, Postfach 6740
D-8700 Würzburg 1

Bitte schicken Sie mir unverbindlich 1 Exemplar der neuen »elektronikpraxis« zum Kennenlernen.

Name

Funktion

Firma

PLZ/Ort

Straße/Postfach

DER NEUE EPSON MX-80 F/T. TYP II. EIN DRUCKER, DER BEIDES KANN: DRUCKEN UND ZEICHNEN. IN AUSGEZEICHNETER QUALITÄT.

Der MX-80 F/T. Typ II.

Der MX-80 F/T, Typ II. Für Einzelblatt, Rollen- und Endlospapier.
9 x 9 Dot Matrix mit Unterlängen. Geschwindigkeit: 80 CPS Bi-Direktional. 96 ASCII,
64 Graphikzeichen. Perfekt in Deutsch (mit Umlauten), Englisch, Französisch,
Amerikanisch. 1 Original und mindestens 3 Kopien. Garantiert. Papierbreite stufenlos
verstellbar von 4" bis 10". Programmierbarer Zeilenabstand mit Tabulator. Druckweg-Optimierung.
Variable Zeichengröße, komprimiert und gespreizt. Kassettenfarbband. Leise. Unter 57 dB.

EPSON

Die Zeit ist da. Für bessere Ideen.

EPSON Deutschland GmbH, Am Seestern 24, Düsseldorf, Tel.: 02 11/59 30 80, Telex: 8 584 786 EPSON ist der führende Hersteller von Mini-Printern.
Weltweit. Mit einem Marktanteil von ca. 70%. EPSON. Ein Tochterunternehmen der SEIKO Group.